Neelima Sandya Chimaladinne – Auction House Capstone

The Auction House API

Project Overview

The Auction House is a .NET 8 Web API project implementing an online auction platform.

Implementation Summary

Task 1: SQLite Database Integration (COMPLETED)

- **Database**: SQLite with Entity Framework Core 8.0
- **Entities**: PortalUser, Asset, Auction, BidHistory with proper relationships
- **Migrations**: Initial schema created with Identity tables
- **Connection**: `Data Source=auctionhouse.db` in appsettings.json

Task 2: Authentication & Authorization (COMPLETED)

- **JWT Authentication**: Full Bearer token implementation
- **User Roles**: Admin, User, Seller, Bidder with role-based policies
- **Registration/Login**: Secure endpoints with password validation
- **Admin Account**: `admin@auctionhouse.com` / `Admin123!` (auto-seeded)
- **Swagger Integration**: JWT Bearer authentication in Swagger UI

Current Project Structure

```
auction-house-capstone-main/
├── dev.md                           # This developer handover document
├── plan.md                          # Implementation plan and status
├── SRS.md                           # System Requirements Specification
├── project_status.md                # Project status documentation
├── TheAuctionHouse.sln              # Solution file
├── test.html                        # Test file
├── .git/                            # Git repository
├── .vscode/                         # VS Code settings
├── Wireframes/                      # UI wireframes
├── Controllers/                     # Legacy controllers (not used)
│
├── TheAuctionHouse/                 # Main Web API Project
│   ├── Controllers/
│   │   └── AuthController.cs        # Authentication & role management
│   ├── Models/
│   │   ├── AuthModels.cs            # DTOs for auth requests/responses
│   │   └── JwtSettings.cs           # JWT configuration model
│   ├── Services/
│   │   ├── IJwtService.cs           # JWT service interface
│   │   └── JwtService.cs          # JWT token generation/validation
│   ├── Properties/
│   │   └── launchSettings.json    # Launch configuration
│   ├── bin/                       # Build output
│   ├── obj/                       # Build intermediate files
│   ├── Program.cs                 # App configuration with JWT & policies
│   ├── appsettings.json           # JWT settings & connection string
│   ├── TheAuctionHouse.csproj     # Project file
│   └── auctionhouse.db            # SQLite database file
│
├── TheAuctionHouse.Domain.Entities/  # Domain Entities Project
│   ├── bin/                       # Build output
│   ├── obj/                       # Build intermediate files
│   ├── PortalUser.cs              # Identity user with wallet properties
│   ├── Asset.cs                   # Asset entity with status enum
│   ├── Auction.cs                 # Auction entity with business logic
│   ├── BidHistory.cs              # Bid tracking entity
│   └── TheAuctionHouse.Domain.Entities.csproj
│
├── TheAuctionHouse.Data.EFCore.SQLite/  # Data Access Project
│   ├── Migrations/                # EF Core migrations
│   │   ├── 20250526095858_InitialCreateAndIdentitySchema.cs
│   │   ├── 20250526095858_InitialCreateAndIdentitySchema.Designer.cs
│   │   └── AuctionHouseDbContextModelSnapshot.cs
│   ├── bin/                       # Build output
│   ├── obj/                       # Build intermediate files
│   ├── Class1.cs                  # AuctionHouseDbContext (renamed from
Class1)
│   └── TheAuctionHouse.Data.EFCore.SQLite.csproj
│
├── TheAuctionHouse.Domain.DataContracts/  # Repository Interfaces
│   ├── bin/                       # Build output
│   ├── obj/                       # Build intermediate files
│   ├── IAppUnitOfWork.cs          # Unit of work pattern interface
│   ├── IAssetRepository.cs        # Asset repository interface
│   ├── IAuctionRepository.cs      # Auction repository interface
│   ├── IPortalUserRepository.cs   # User repository interface
│   ├── IRepository.cs             # Generic repository interface
│   └── TheAuctionHouse.Domain.DataContracts.csproj
│
├── TheAuctionHouse.Domain.ServiceContracts/  # Service Interfaces & DTOs
│   ├── DataTransferObjects/       # Legacy DTOs (not currently used)
│   │   ├── AssetInformationUpdateRequest.cs
│   │   ├── AssetResponse.cs
│   │   ├── AuctionResponse.cs
│   │   ├── BidHistoryResponse.cs
│   │   ├── ForgotPasswordRequest.cs
│   │   ├── LoginRequest.cs
│   │   ├── PostAuctionRequest.cs
│   │   ├── ResetPasswordRequest.cs
│   │   ├── SignUpRequest.cs
│   │   ├── WalletBalenceResponse.cs
│   │   └── WalletTransactionRequest.cs
│   ├── bin/                        # Build output
│   ├── obj/                        # Build intermediate files
│   ├── IAssetService.cs            # Asset service interface
│   ├── IAuctionService.cs          # Auction service interface
│   ├── IPortalUserService.cs       # User service interface
│   ├── IWalletService.cs           # Wallet service interface
│   └── TheAuctionHouse.Domain.ServiceContracts.csproj
│
├── TheAuctionHouse.Domain.Services/  # Service Implementations (Legacy)
│   ├── bin/                        # Build output
│   ├── obj/                        # Build intermediate files
│   ├── AssetService.cs             # Legacy asset service
│   ├── PortalUserService.cs        # Legacy user service
│   └── TheAuctionHouse.Domain.Services.csproj
│
├── TheAuctionHouse.Common/         # Common Utilities
│   ├── bin/                        # Build output
│   ├── obj/                        # Build intermediate files
│   ├── Error.cs                    # Error handling classes
│   ├── IEmailService.cs            # Email service interface
│   ├── Result.cs                   # Result pattern implementation
│   ├── StandardErrors.cs           # Standard error definitions
│   ├── ValidationHelper.cs        # Validation utilities
│   └── TheAuctionHouse.Common.csproj
│
├── TheAuctionHouse.Data.EFCore.InMemory/  # Legacy InMemory Data (Not Used)
│   └── (Legacy project - replaced by SQLite)
│
└── TheAuctionHouse.Domain.Services.Tests/  # Test Project
    └── (Test files for domain services)
```

Authentication & Authorization Implementation Details

JWT Configuration

```
"JwtSettings": {
  "SecretKey": "YourSuperSecretKeyThatIsAtLeast32CharactersLongForSecurity!",
  "Issuer": "TheAuctionHouseAPI",
  "Audience": "TheAuctionHouseUsers",
  "ExpirationInMinutes": 60
}
```

User Roles & Policies

- **Admin**: Full system access, user/role management
- **User**: Default role for all registrations
- **Seller**: Can create and manage auctions
- **Bidder**: Can place bids on auctions

Authorization Policies (Configured in Program.cs)

```
"AdminOnly" => RequireRole(UserRoles.Admin)
"UserOrAdmin" => RequireRole(UserRoles.User, UserRoles.Admin)
"SellerOrAdmin" => RequireRole(UserRoles.Seller, UserRoles.Admin)
"BidderOrAdmin" => RequireRole(UserRoles.Bidder, UserRoles.Admin)
"AuthenticatedUser" => RequireAuthenticatedUser()
```

Available Auth Endpoints

- `POST /api/auth/register` - User registration (assigns User role)
- `POST /api/auth/login` - Login with JWT token response
- `GET /api/auth/me` - Get current user info [Authorize]
- `POST /api/auth/assign-role` - Assign role [AdminOnly]
- `POST /api/auth/remove-role` - Remove role [AdminOnly]
- `GET /api/auth/roles` - List all roles [AdminOnly]
- `GET /api/auth/users` - List all users with roles [AdminOnly]

Entity Relationships & Business Rules

Asset Entity

```
public class Asset
{
    public int Id { get; set; }
    public string OwnerId { get; set; } // FK to PortalUser.Id
    public string Title { get; set; }
    public string Description { get; set; }
    public int RetailValue { get; set; }
    public AssetStatus Status { get; set; }

    // Navigation properties
    public virtual PortalUser Owner { get; set; }
    public virtual ICollection<Auction> Auctions { get; set; }
}

public enum AssetStatus
{
    Draft,            // Can be edited
    OpenToAuction,    // Ready for auction
    ClosedForAuction // Currently in auction or sold
}
```

Auction Entity

```
public class Auction
{
    public int Id { get; set; }
    public string SellerId { get; set; } // FK to PortalUser.Id
    public int AssetId { get; set; }
    public int ReservedPrice { get; set; }
    public decimal CurrentHighestBid { get; set; }
    public string? CurrentHighestBidderId { get; set; } // FK to
PortalUser.Id
    public int MinimumBidIncrement { get; set; }
    public DateTime StartDate { get; set; }
    public int TotalMinutesToExpiry { get; set; }
    public AuctionStatus Status { get; set; }

    // Business logic methods
    public int GetRemainingTimeInMinutes()
    public bool IsExpired()
    public bool IsExpiredWithoutBids()
    public bool IsLive()
}
```

PortalUser Entity (Identity Integration)

```
public class PortalUser : IdentityUser
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public decimal WalletBalance { get; set; }
    public decimal BlockedAmount { get; set; }

    // Navigation properties
    public virtual ICollection<Asset> AssetsOwned { get; set; }
    public virtual ICollection<Auction> AuctionsListedAsSeller { get; set; }
    public virtual ICollection<BidHistory> BidsPlaced { get; set; }
}
```

Task 3: Assets Management Implementation Guide

Required DTOs (Create in Models/)

```
// Models/AssetModels.cs
public class CreateAssetRequest
{
    [Required, StringLength(150, MinimumLength = 10)]
    public string Title { get; set; }

    [Required, StringLength(1000, MinimumLength = 10)]
    public string Description { get; set; }

    [Required, Range(1, int.MaxValue)]
    public int RetailValue { get; set; }
}

public class UpdateAssetRequest
{
    [Required, StringLength(150, MinimumLength = 10)]
    public string Title { get; set; }
    [Required, StringLength(1000, MinimumLength = 10)]
    public string Description { get; set; }

    [Required, Range(1, int.MaxValue)]
    public int RetailValue { get; set; }
}

public class AssetResponse
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public int RetailValue { get; set; }
    public string Status { get; set; }
    public string OwnerName { get; set; }
    public DateTime CreatedDate { get; set; }
}
```

Assets Controller Implementation

```
// Controllers/AssetsController.cs
[ApiController]
[Route("api/[controller]")]
[Authorize] // All endpoints require authentication
public class AssetsController : ControllerBase
{
    private readonly AuctionHouseDbContext _context;
    private readonly UserManager<PortalUser> _userManager;

    // Required endpoints based on SRS.md:

    [HttpPost] // Create Asset (4.1.1)
    [Authorize(Policy = "UserOrAdmin")]

    [HttpPut("{id}")] // Update Asset (4.1.2)
    [Authorize(Policy = "UserOrAdmin")]

    [HttpPatch("{id}/status")] // Change status (4.1.3)
    [Authorize(Policy = "UserOrAdmin")]

    [HttpDelete("{id}")] // Delete Asset (4.1.7)
    [Authorize(Policy = "UserOrAdmin")]

    [HttpGet] // Get user's assets (4.1.8)
    [Authorize(Policy = "UserOrAdmin")]

    [HttpGet("{id}")] // Get specific asset
    [Authorize(Policy = "AuthenticatedUser")]
}
```

Business Rules to Implement (from SRS.md)

1. **Asset Creation (4.1.1)**
   - Title: 10-150 chars, no special chars, trim spaces
   - Description: 10-1000 chars, allow special chars
   - RetailValue: positive integer
   - Default status: Draft
   - Owner: Current authenticated user
2. **Asset Updates (4.1.2)**
   - Only Draft status assets can be updated
   - Same validation as creation
   - Only asset owner can update
3. **Status Management**
   - Draft → OpenToAuction (user action)
   - OpenToAuction → ClosedForAuction (system when auction starts)
   - ClosedForAuction → OpenToAuction (system when auction ends without bids)
4. **Authorization Rules**
   - Users can only manage their own assets
   - Admins can view/manage all assets
   - Assets in ClosedForAuction cannot be deleted

Task 4: Wallet Management Implementation Guide

Required DTOs

```
// Models/WalletModels.cs
public class DepositRequest
{
    [Required, Range(1, 999999)]
    public decimal Amount { get; set; }
}

public class WithdrawRequest
{
    [Required, Range(1, 999999)]
    public decimal Amount { get; set; }
}

public class WalletResponse
{
    public decimal WalletBalance { get; set; }
    public decimal BlockedAmount { get; set; }
    public decimal AvailableBalance => WalletBalance - BlockedAmount;
    public List<BlockedAmountDetail> BlockedAmounts { get; set; }
}
```

```
public class BlockedAmountDetail
{
    public int AuctionId { get; set; }
    public string AssetTitle { get; set; }
    public decimal BidAmount { get; set; }
    public DateTime BidDate { get; set; }
}
```

Wallet Controller Implementation

```
// Controllers/WalletController.cs
[ApiController]
[Route("api/[controller]")]
[Authorize] // All endpoints require authentication
public class WalletController : ControllerBase
{
    // Required endpoints based on SRS.md:

    [HttpPost("deposit")] // Deposit money (4.2.1)
    [Authorize(Policy = "UserOrAdmin")]

    [HttpPost("withdraw")] // Withdraw money (4.2.2)
    [Authorize(Policy = "UserOrAdmin")]

    [HttpGet] // Get wallet dashboard (4.2.3)
    [Authorize(Policy = "UserOrAdmin")]
}
```

Business Rules to Implement

1. **Deposit (4.2.1)**
   - Amount: 1-999,999 (positive integer)
   - Add to user’s WalletBalance
   - Log transaction for audit
2. **Withdrawal (4.2.2)**
   - Amount: 1-999,999 (positive integer)
   - Check available balance (WalletBalance - BlockedAmount)
   - Prevent withdrawal if insufficient funds
   - Subtract from WalletBalance
3. **Balance Blocking (Business Rule 1)**
   - Block bid amount when user places highest bid
   - Unblock when another user outbids
   - Transfer blocked amount on auction completion

Getting Current User in Controllers

```
// Get current user ID from JWT token
private string GetCurrentUserId()
{
    return User.FindFirst(ClaimTypes.NameIdentifier)?.Value ?? string.Empty;
}

// Get current user entity
private async Task<PortalUser?> GetCurrentUserAsync()
{
    var userId = GetCurrentUserId();
    return await _userManager.FindByIdAsync(userId);
}
```

Database Context Usage

```
// Inject AuctionHouseDbContext in controllers
private readonly AuctionHouseDbContext _context;

// Example queries
var userAssets = await _context.Assets
    .Where(a => a.OwnerId == userId)
    .Include(a => a.Owner)
    .ToListAsync();

var availableAssets = await _context.Assets
    .Where(a => a.Status == AssetStatus.OpenToAuction)
    .Include(a => a.Owner)
    .ToListAsync();
```

Testing Strategy

Manual Testing in Swagger

1. **Authentication Flow**
   - Register new user → Login → Copy JWT token
   - Use “Authorize” button in Swagger UI
   - Test protected endpoints
2. **Asset Management Testing**
   - Create assets with different users
   - Test validation rules
   - Test status transitions
   - Test authorization (users can only manage own assets)
3. **Wallet Testing**
   - Test deposit/withdrawal with validation
   - Test insufficient funds scenarios
   - Verify balance calculations

Integration Testing

- Test asset creation → auction posting → bidding → wallet blocking
- Test role-based access control
- Test business rule enforcement

Important Notes

- **JWT tokens contain user roles** - use for authorization
- **Entity relationships are configured** - use Include() for navigation properties
- **Database migrations are set up** - run `dotnet ef database update` after changes
- **Admin account exists** - use for testing admin-only features
- **Swagger is configured** - use for API testing and documentation

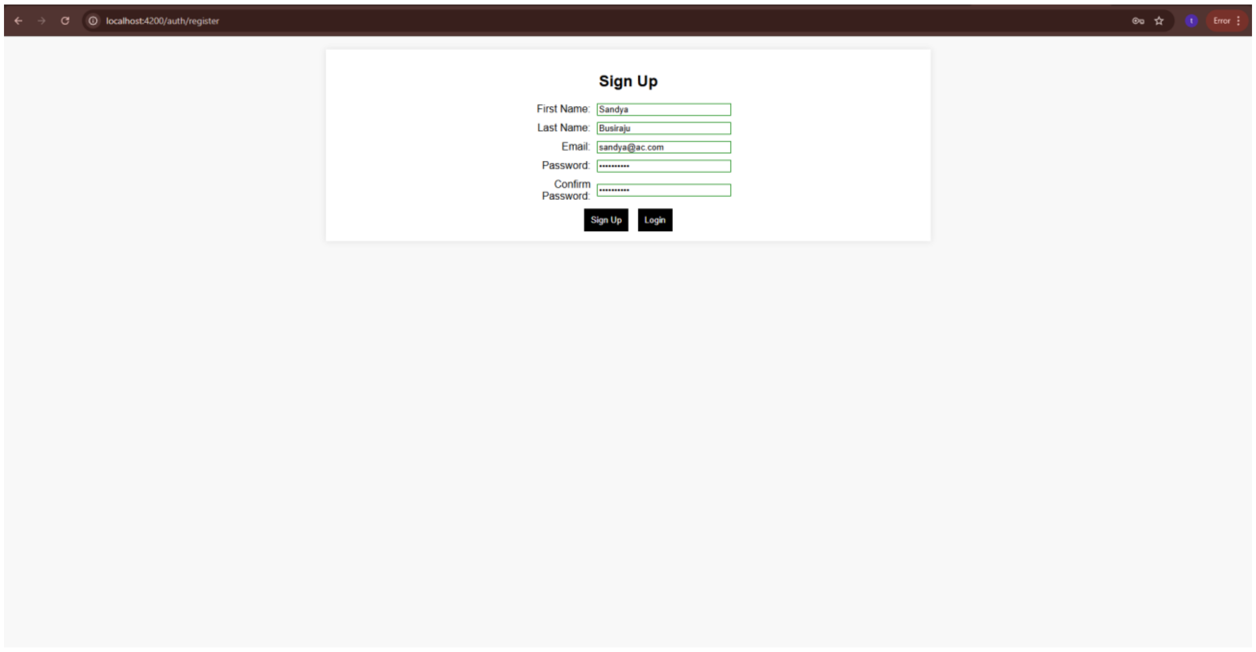
Running the Application

```
cd TheAuctionHouse
dotnet run --urls "http://localhost:5000"
```

Access Swagger UI at: `http://localhost:5000`

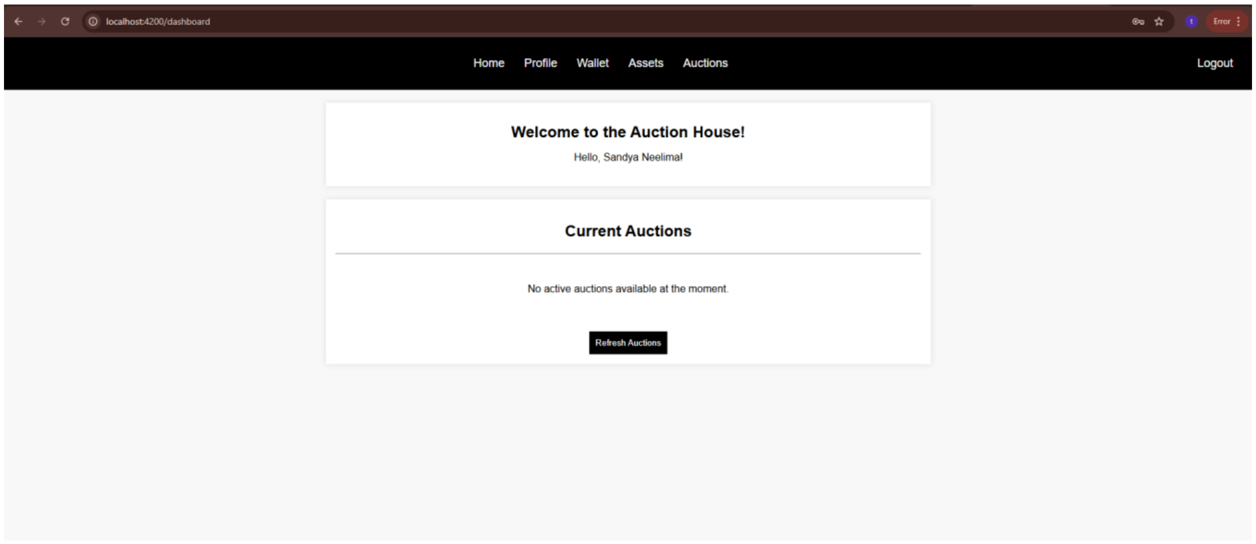
Default admin credentials: - Email: `admin@auctionhouse.com` - Password: `Admin123!`

Screenshots


localhost:4200/auth/register

Sign Up

First Name:	Sandya
Last Name:	Busiraju
Email:	sandya@ac.com
Password:	••••••••••
Confirm Password:	••••••••••

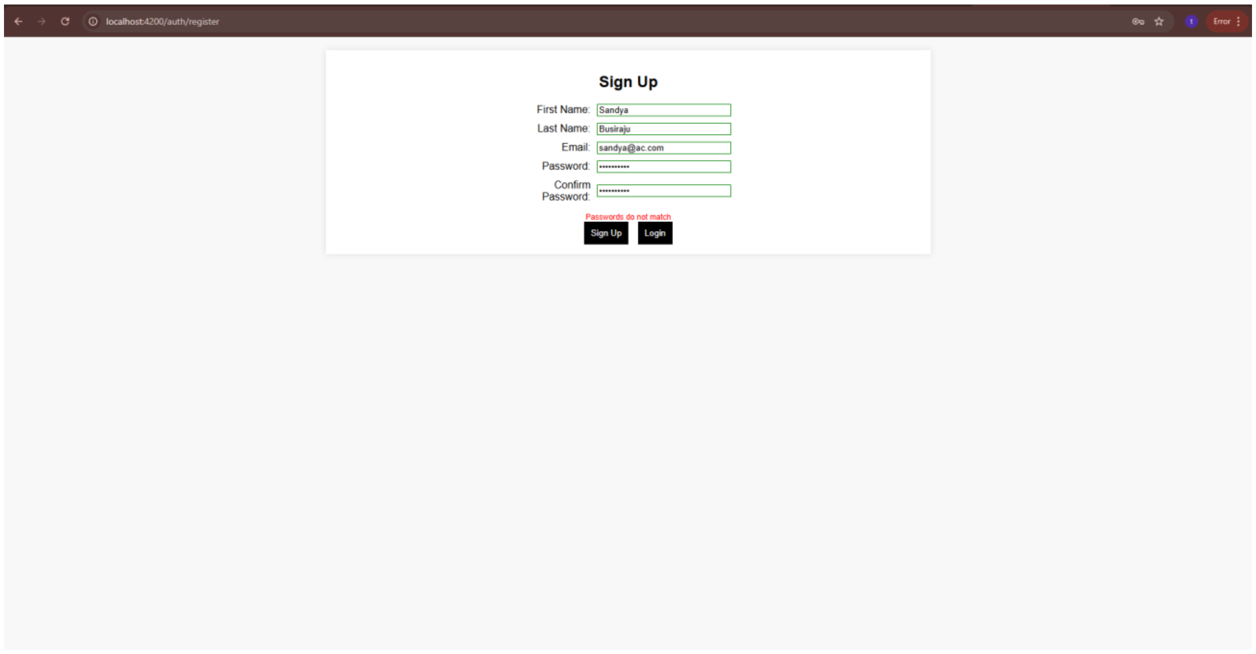

localhost:4200/dashboard

Welcome to the Auction House!

Hello, Sandya Neelima!

Current Auctions

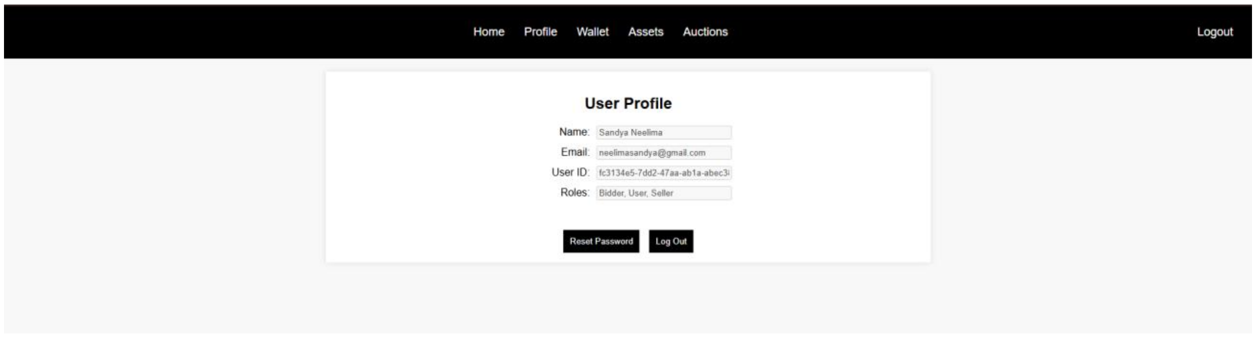
No active auctions available at the moment.


localhost:4200/auth/register

Sign Up

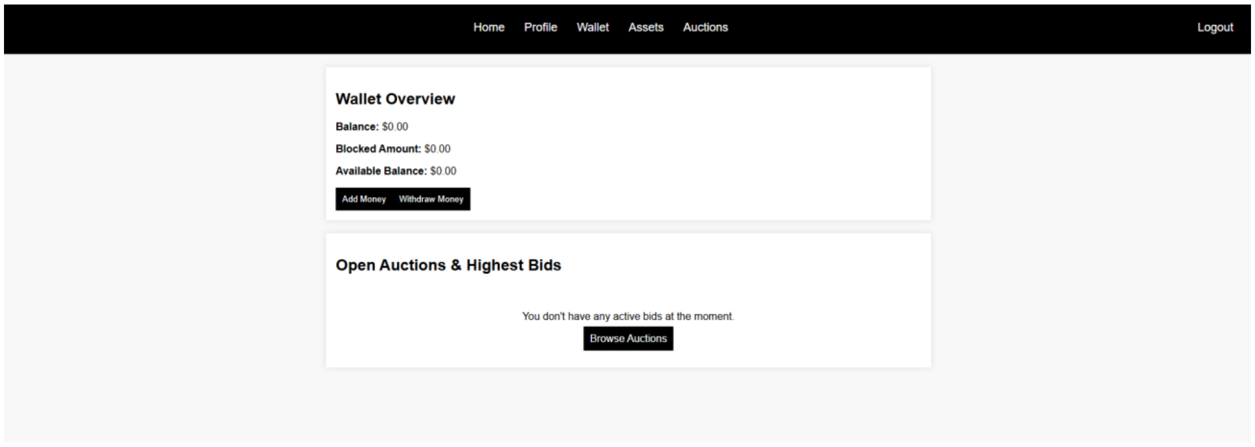
First Name:	Sandya
Last Name:	Busiraju
Email:	sandya@ac.com
Password:	••••••••••
Confirm Password:	••••••••••

Passwords do not match



User Profile

Name:	Sandya Neelima
Email:	neelimasandya@gmail.com
User ID:	fc3134e5-7dd2-47aa-ab1a-abec3
Roles:	Bidder, User, Seller



Wallet Overview

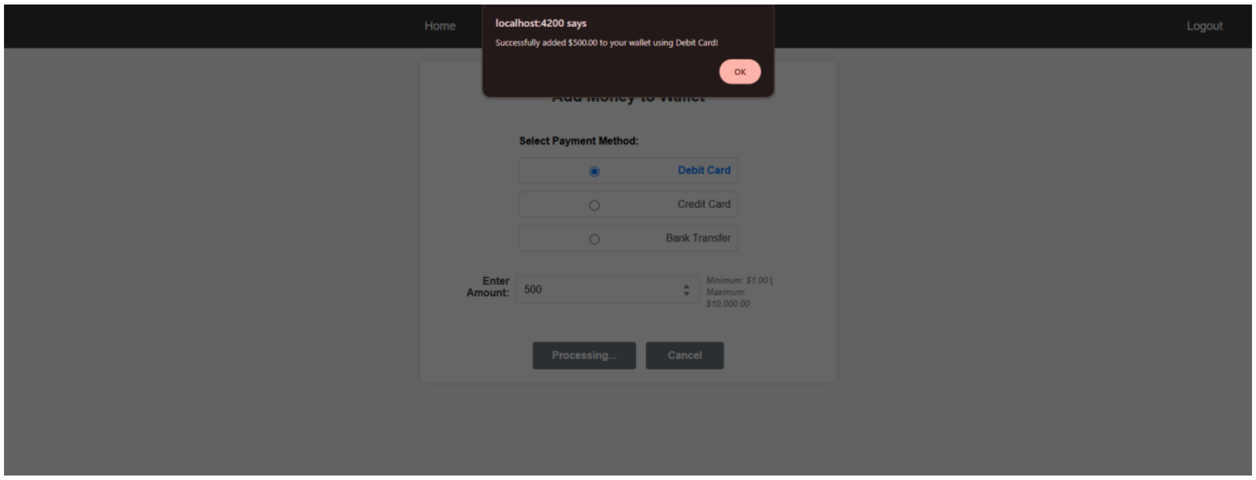
Balance: $0.00

Blocked Amount: $0.00

Available Balance: $0.00

Open Auctions & Highest Bids

You don't have any active bids at the moment.



localhost:4200 says

Successfully added $500.00 to your wallet using Debit Card!

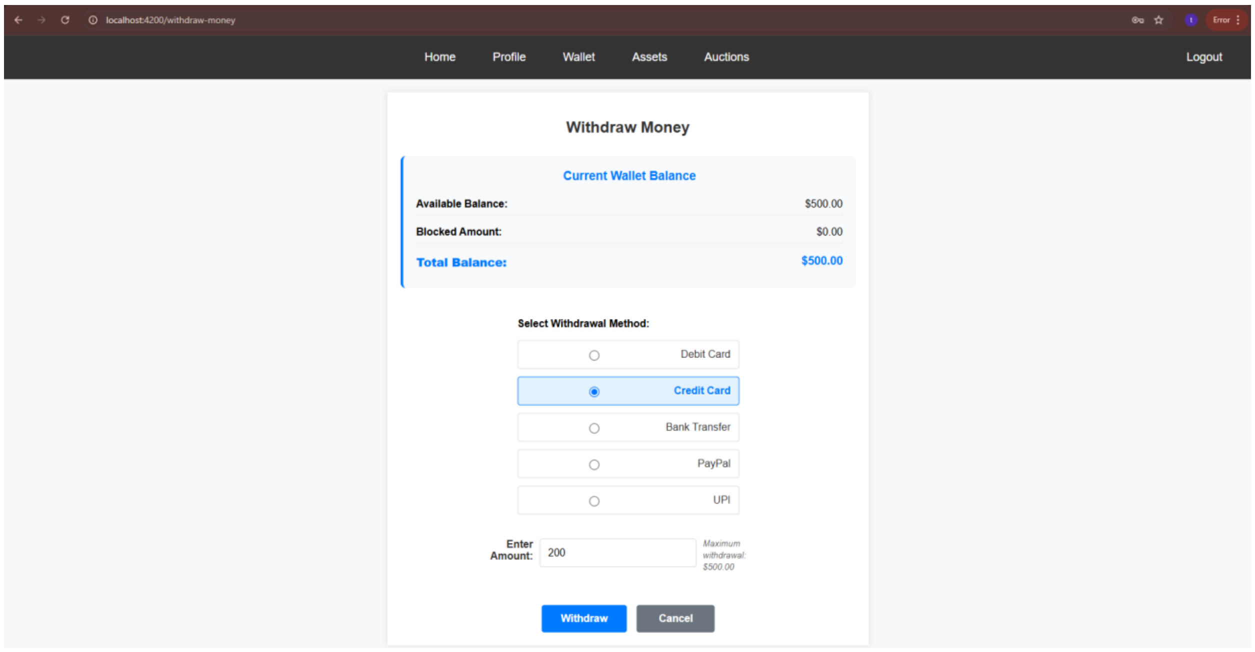
Add Money to Wallet

Select Payment Method:

[✓ Debit Card]
Credit Card
Bank Transfer

Enter Amount: 500

Minimum: $1.00 | Maximum: $10,000.00


localhost:4200/withdraw-money

Home Profile Wallet Assets Auctions Logout

Withdraw Money

Current Wallet Balance

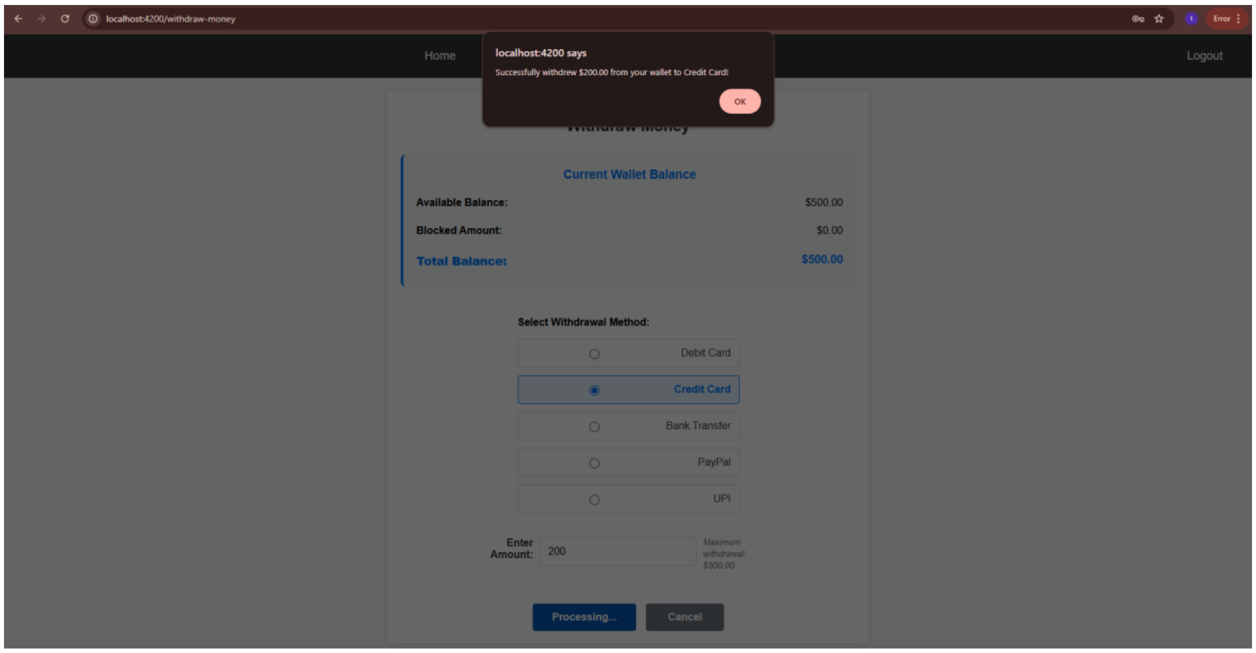
Available Balance:	$500.00
Blocked Amount:	$0.00
Total Balance:	$500.00

Select Withdrawal Method:

Debit Card
[✓ Credit Card]
Bank Transfer
PayPal
UPI

Enter Amount: 200 *Maximum withdrawal: $500.00*

Withdraw Cancel


localhost:4200/withdraw-money

localhost:4200 says

Successfully withdrew $200.00 from your wallet to Credit Card!

OK

Current Wallet Balance

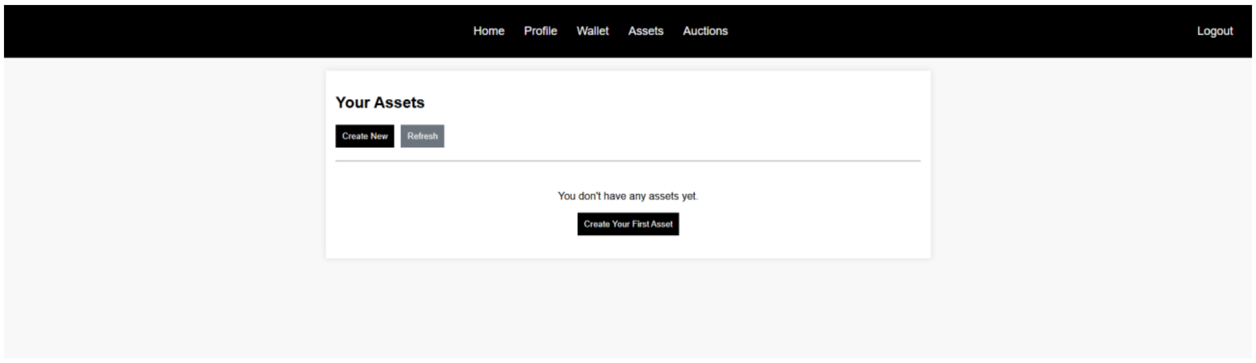
Available Balance:	$500.00
Blocked Amount:	$0.00
Total Balance:	$500.00

Select Withdrawal Method:

Debit Card
[✓ Credit Card]
Bank Transfer
PayPal
UPI

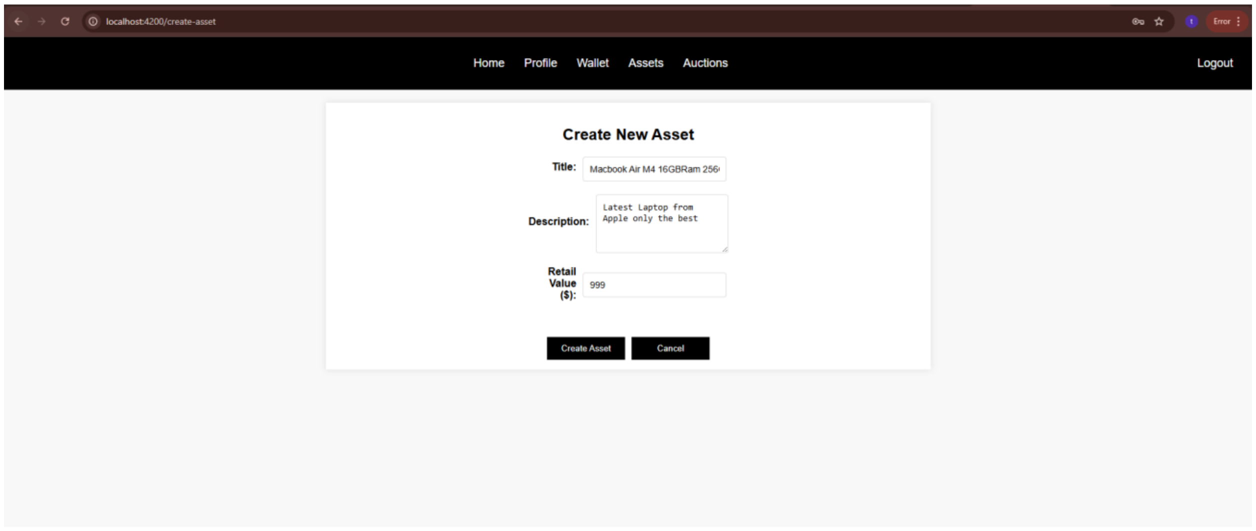
Enter Amount: 200 *Maximum withdrawal: $500.00*

Processing... Cancel



Your Assets

You don't have any assets yet.

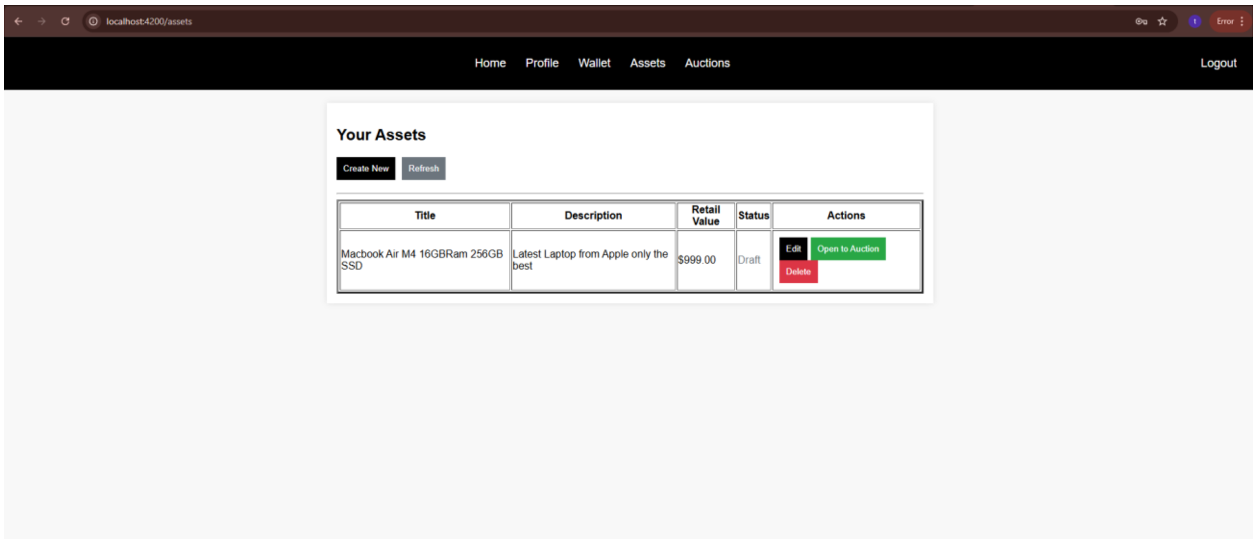


Create New Asset

Title: Macbook Air M4 16GBRam 256

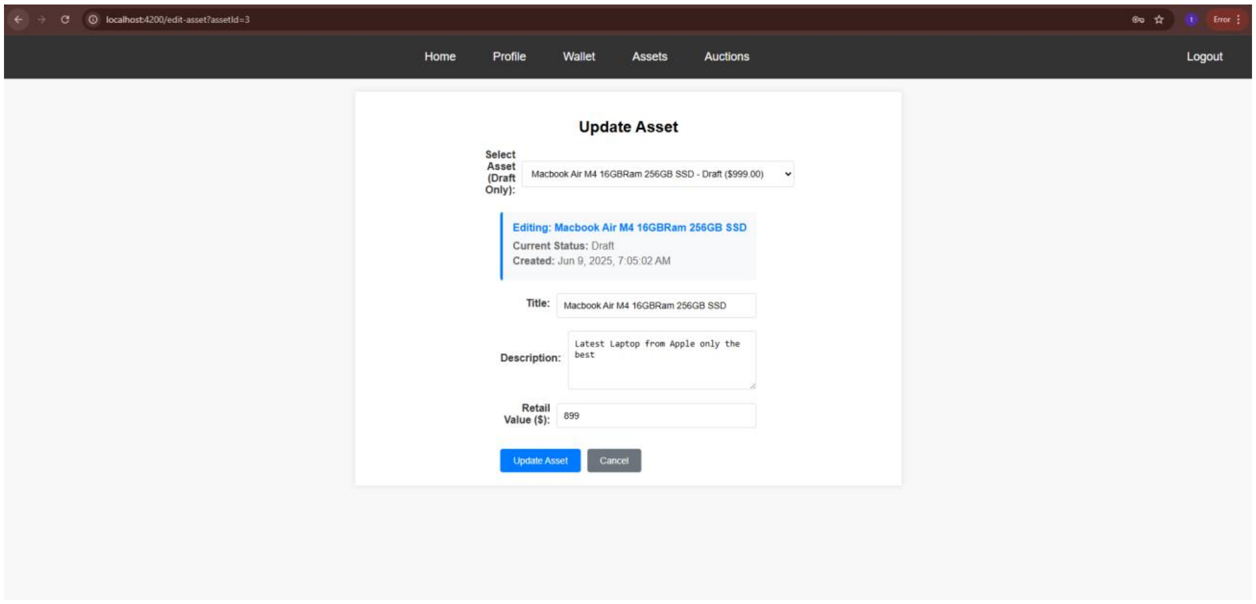
Description: Latest Laptop from Apple only the best

Retail Value ($): 999



Your Assets

Title	Description	Retail Value	Status	Actions
Macbook Air M4 16GBRam 256GB SSD	Latest Laptop from Apple only the best	$999.00	Draft	Edit Open to Auction Delete

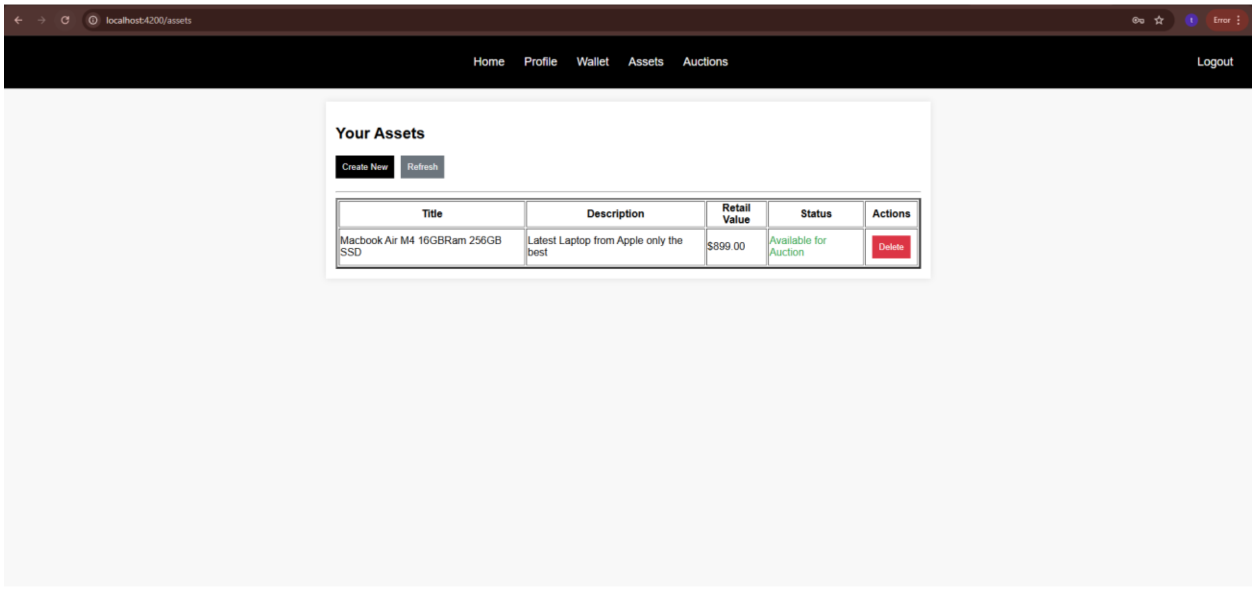


Update Asset

Select Asset (Draft Only):	Macbook Air M4 16GBRam 256GB SSD - Draft ($999.00)

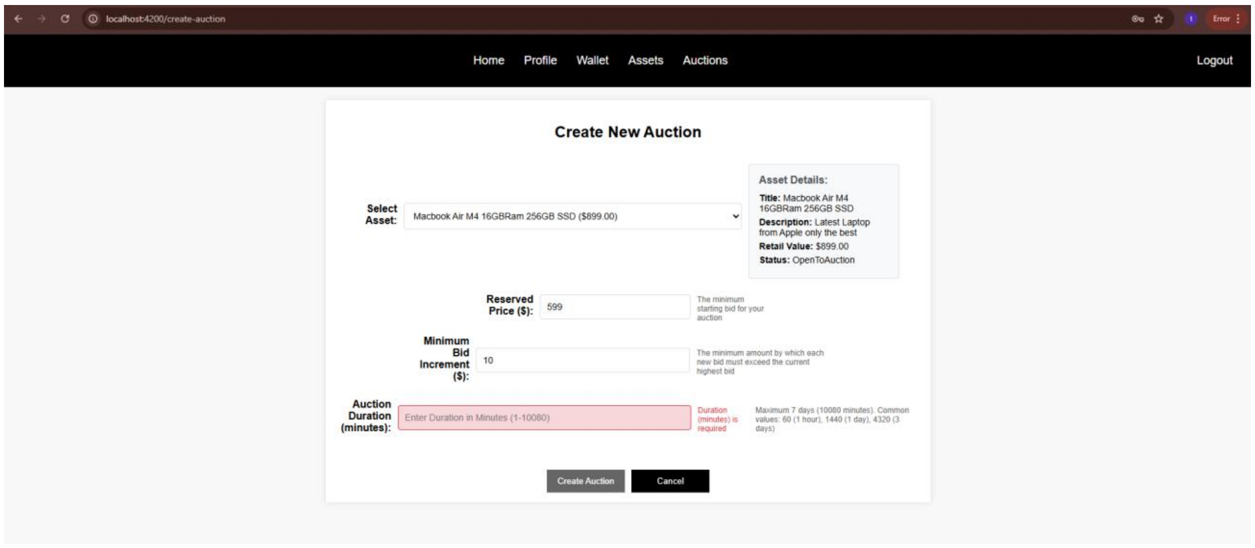
Editing: Macbook Air M4 16GBRam 256GB SSD
Current Status: Draft
Created: Jun 9, 2025, 7:05:02 AM

Title:	Macbook Air M4 16GBRam 256GB SSD
Description:	Latest Laptop from Apple only the best
Retail Value ($):	899



Your Assets

Title	Description	Retail Value	Status	Actions
Macbook Air M4 16GBRam 256GB SSD	Latest Laptop from Apple only the best	$899.00	Available for Auction	Delete


localhost:4200/create-auction

Home Profile Wallet Assets Auctions Logout

Create New Auction

Select Asset: Macbook Air M4 16GBRam 256GB SSD ($899.00)

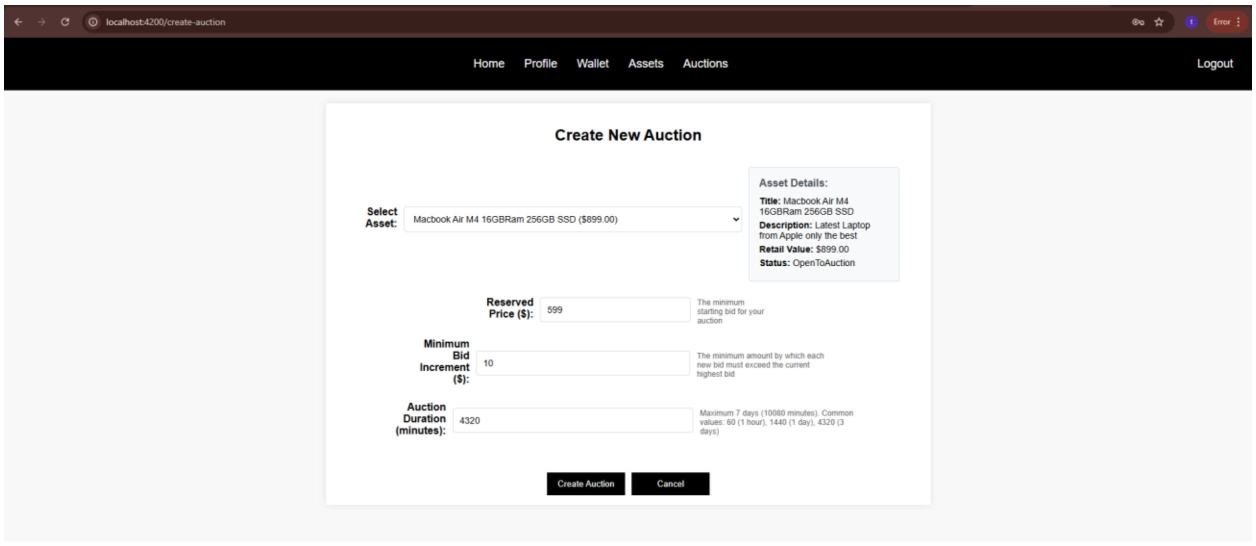
Asset Details:

Title: Macbook Air M4 16GBRam 256GB SSD
Description: Latest Laptop from Apple only the best
Retail Value: $899.00
Status: OpenToAuction

Reserved Price ($): 599 — The minimum starting bid for your auction

Minimum Bid Increment ($): 10 — The minimum amount by which each new bid must exceed the current highest bid

Auction Duration (minutes): Enter Duration in Minutes (1-10080) — Duration (minutes) is required — Maximum 7 days (10080 minutes). Common values: 60 (1 hour), 1440 (1 day), 4320 (3 days)


localhost:4200/create-auction

Home Profile Wallet Assets Auctions Logout

Create New Auction

Select Asset: Macbook Air M4 16GBRam 256GB SSD ($899.00)

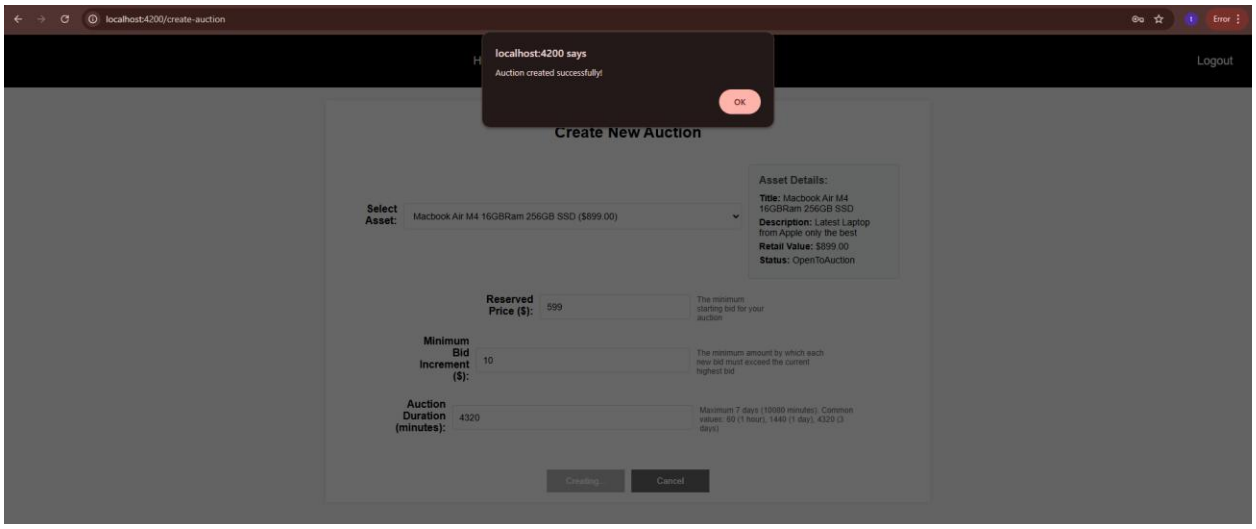
Asset Details:

Title: Macbook Air M4 16GBRam 256GB SSD
Description: Latest Laptop from Apple only the best
Retail Value: $899.00
Status: OpenToAuction

Reserved Price ($): 599 — The minimum starting bid for your auction

Minimum Bid Increment ($): 10 — The minimum amount by which each new bid must exceed the current highest bid

Auction Duration (minutes): 4320 — Maximum 7 days (10080 minutes). Common values: 60 (1 hour), 1440 (1 day), 4320 (3 days)



localhost:4200/create-auction

localhost:4200 says

Auction created successfully!

OK

Logout

Create New Auction

Select Asset: Macbook Air M4 16GBRam 256GB SSD ($899.00)

Asset Details:

Title: Macbook Air M4 16GBRam 256GB SSD

Description: Latest Laptop from Apple only the best

Retail Value: $899.00

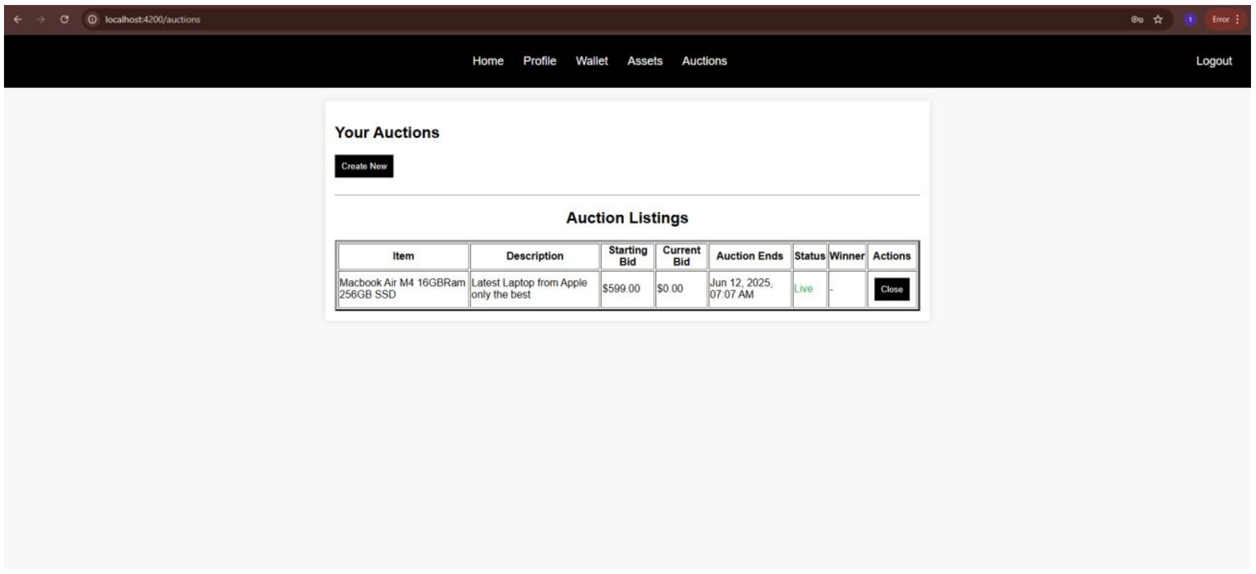
Status: OpenToAuction

Reserved Price ($): 599 — The minimum starting bid for your auction

Minimum Bid Increment ($): 10 — The minimum amount by which each new bid must exceed the current highest bid

Auction Duration (minutes): 4320 — Maximum 7 days (10080 minutes). Common values: 60 (1 hour), 1440 (1 day), 4320 (3 days)

Creating... Cancel



localhost:4200/auctions

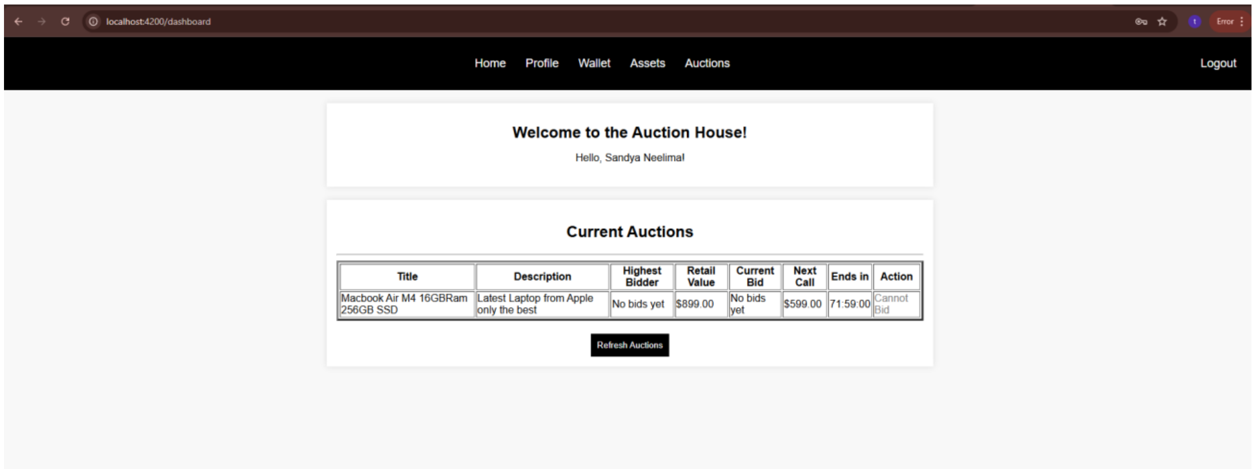
Home Profile Wallet Assets Auctions Logout

Your Auctions

Create New

Auction Listings

Item	Description	Starting Bid	Current Bid	Auction Ends	Status	Winner	Actions
Macbook Air M4 16GBRam 256GB SSD	Latest Laptop from Apple only the best	$599.00	$0.00	Jun 12, 2025, 07:07 AM	Live	-	Close



localhost:4200/dashboard

Home Profile Wallet Assets Auctions Logout

Welcome to the Auction House!

Hello, Sandya Neelima!

Current Auctions

Title	Description	Highest Bidder	Retail Value	Current Bid	Next Call	Ends in	Action
Macbook Air M4 16GBRam 256GB SSD	Latest Laptop from Apple only the best	No bids yet	$899.00	No bids yet	$599.00	71:59:00	Cannot Bid

Refresh Auctions