# **AfriBase - African Artifact Auction Website**



## **Web Development 2B (WE22B) Student Group:** Masase Mukhindi **Student Number:** 23412220 **Date:** 2025/04/28 **Assessment:** SS2 (Execution)

## **Table of Contents**

* Task 1: Understanding ASP.NET Core MVC Architecture for AfriBase Project  
  + Introduction
  + Model Component
  + View Component
  + Controller Component
  + Component Interaction
  + Benefits for AfriBase Project
* Task 2: AfriBase Web Application  
  + 1. Project Overview
  + 2. Key Features
    - 2.1 User Authentication & Management
    - 2.2 Artifact Management
    - 2.3 Bidding System
    - 2.4 Database Integration
    - 2.5 African-Themed UI Design
  + 3. Technology Stack
  + 4. Project Structure
  + 5. Setup Instructions
  + 6. Key Files and Components
  + 7. Features Implementation Details
  + 8. Future Enhancements
  + 9. Conclusion
  + 10. Flowchart
  + 11.Page Design
  + 12. References

# **Task 1: Understanding ASP.NET Core MVC Architecture for AfriBase Project:**

The AfriBase Auctions web platform needs to implement a powerful system which combines artifact management with user authorization features alongside bidding and artwork addition functions. Applicability of the ASP.NET Core MVC platform emerges from its choreographed format and clearly divided components and numerous built-in features. The Model View Controller system in ASP.NET Core MVC matches the specifications of AfriBase project requirements.

## **Model Component:**

The Model component of the AfriBase project contains both data structures with application logic and business rules. The Model component contains all essential functions involving artifacts and their management with users and bid section as well as the authentication aspect.The AfriBase Project contains three essential models that form its foundation.

### **Key Models in the AfriBase Project:**

1. **User/ApplicationUser Model**
   * The design adopts and expands the capabilities of the ASP.NET Core Identity framework.
   * The model contains the fields for UserId, Username, Email, PasswordHash, RegistrationDate, and Role
   * Handles user authentication, authorization, and profile management
   * The system includes automated validation procedures which ensure correct data integrity.
2. **Artifact Model**
   * The artifact model comprises ArtifactId as well as Name, Description, Origin, EstimatedYear, EstimatedValue, Category, Condition, ImageUrl, AddedDate and OwnerId.
   * The database maintains relationships through User and Bids with Owner
   * Implements business logic for artifact validation and value calculation
3. **Bid Model**
   * This model contains six fields: BidId along with UserId, ArtifactId, BidAmount, BidDate, and Status.
   * The system requires User and Artifact entities to build its relationships.
   * The model includes code for validating and managing bid status operations.
4. **ViewModels**
   * LoginViewModel: Handles login form data
   * ArtifactDetailViewModel combines information about artifacts alongside bids which relate to them
   * BidViewModel: Manages form data for new bids

5. **Data Access Layer:**

* The ApplicationDbContext component creates a connection to the database through Entity Framework Core.
* Repository Pattern: Organizes data access for each entity
* DbInitializer serves to create database initialization data that facilitates testing and demo purposes.

## **View Component:**

The View component in the AfriBase project is responsible for the user interface, focusing on both functionality and African-inspired aesthetics.

### **Key Views in the AfriBase Project:**

1. **Authentication Views**
   * Login.cshtml: Provides user login functionality
   * Register.cshtml: Handles new user registration
   * ForgotPassword.cshtml: Manages password recovery
2. **Artifact Views**
   * Index.cshtml: Displays artifact catalog with filtering
   * Details.cshtml: Shows detailed artifact information and bidding interface
   * Create.cshtml: Offers form for submitting new artifacts (admin/staff only)
   * Edit.cshtml: Allows updating artifact information (admin/staff only)
3. **Bidding Views**
   * BidHistory.cshtml: Presents bidding history for artifacts
   * PlaceBid.cshtml: Provides interface for submitting bids
4. **Dashboard Views**
   * UserDashboard.cshtml: Displays user-specific information
   * AdminDashboard.cshtml: Provides system management tools
5. **Shared Views**
   * \_Layout.cshtml: Defines main layout with African-inspired design
   * \_LoginPartial.cshtml: Shows authentication status
   * \_ValidationScriptsPartial.cshtml: Includes validation scripts

### **UI/UX Features:**

* Responsive design using Bootstrap for mobile compatibility
* African-inspired color schemes (earth tones, vibrant accents)
* Cultural typography and imagery reflecting African heritage
* Consistent navigation and user experience throughout the application

## **Controller Component:**

Through the Controller component users can request data processing operations and view selection determination between Model and View.

### **Key Controllers in the AfriBase Project:**

1. **AccountController**
   * The component offers control features for Login, Register, Logout, ForgotPassword and ResetPassword functions
   * Implements security measures for CSRF protection
   * The user management features within the application utilize ASP.NET Core Identity.
   * Handles user login state and security
2. **ArtifactController**
   * The controller supports several processes which include Index, Details, Create, Edit and Delete.
   * The system applies different permissions to users depending on their assigned roles in the system.
   * The database contains a system that controls artifact lifecycle management.
   * Handles artifact image uploads
3. **BidController**
   * Supports actions such as PlaceBid, BidHistory, and UserBids
   * The system implements rules for placing bids by ensuring user status requirements and minimum bid increments.
   * Records all bid activities
   * Manages bid status and notifications
4. **HomeController**
   * Handles actions like Index, About, and Contact
   * Displays landing page and general information
   * Users can access the main system functions through this component.

### **Controller Responsibilities:**

* Input validation along with model binding processes operate in the system
* Handling errors appropriately
* Checking authentication and authorization
* Managing transactions for data integrity
* Security auditing functions are handled through logging operations during system activities.
* Selecting appropriate views and redirections

## **Component Interaction:**

Through its architecture MVC provides distinct areas of focus and helps different components communicate efficiently.

**User Authentication Flow:**

1. **User Authentication Flow**:  
   * The user presents login credentials through the Login view.
   * AccountController validates credentials
   * ApplicationUser model verifies against database
   * The controller proceeds to the correct dashboard after validation or shows detected errors to the user.
2. **Artifact Listing Flow**:  
   * The artifact catalog becomes available to users through URL requests.
   * ArtifactController queries database
   * Data retrieval and data return functions occur through the Artifact model.
   * Index view shows a list of artifacts through an interface which follows an African design theme.
3. **Bidding Process Flow**:  
   * The PlaceBid view enables users to submit their bids through it.
   * BidController conducts a process to ensure both the value of submitted bids and the eligibility of users.
   * Bid model creates database record
   * The controller will either display the artifact details update or present validation errors to the user.

## **Benefits for AfriBase Project:**

1. **Scalability**: The MVC architecture allows for easy expansion of features and scaling of the application as AfriBase grows.
2. **Maintainability**: Clear separation of concerns makes the codebase easier to maintain and update over time.
3. **Security**: Built-in features for authentication, authorization, and data validation enhance system security.
4. **Testability**: The modular nature of MVC facilitates unit testing of individual components.
5. **Cultural Integration**: The View layer can incorporate African design elements while maintaining functional separation.

By leveraging ASP.NET Core MVC architecture, the AfriBase project benefits from a structured approach to building a complex web application while maintaining flexibility, security, and cultural authenticity.

**Task 2: AfriBase Web Application:**

## **1. Project Overview:**

The AfriBase platform operates as an ASP.NET Core MVC web application which focuses on online auction management for authentic African artifacts. The application allows users to navigate artifacts and conduct online bidding through its design interface that enforces consistent African design principles (Microsoft, 2024).

Modern technology and African cultural heritage unite through this application platform to protect genuine African artifacts while allowing their responsible trading and appreciation.

## **2. Key Features:**

### **2.1 User Authentication & Management:**

The application implements an authentication system which ensures safe user access and controlled roles at all points within the application.

Two different entry portals enable access by standard users alongside administrators

* **Separate Login Portals**: Distinct entry points for standard users and administrators
* **User Registration with Validation**:
  + Email format validation
  + Password complexity requirements
  + Duplicate account prevention
* **Password Recovery Functionality**:
  + Security question option
  + Email-based recovery process
* **User Profile Management**:
  + Personal information updates
  + Contact details management
  + Password changes
* **Role-Based Authorization** (Microsoft, 2024):
  + Admin role with full system access
  + User role with appropriate restrictions
  + Authorization attributes for access control
* **Session Management**:
  + Secure session tokens
  + Timeout handling
  + Protection against cross-site request forgery

### 

### **2.2 Artifact Management:**

The artifact system maintains complete tools for digital African cultural item management through its system.

* **Complete CRUD Operations**:
  + Create new artifact listings with detailed information
  + Read/retrieve artifact details with efficient filtering
  + Update existing artifact information as needed
  + Delete artifacts with proper authorization checks
* **Image Upload and Management**:
  + Multiple image support for detailed viewing
  + Thumbnail generation for catalog display
  + Secure file handling to prevent malicious uploads
* **Detailed Artifact Information Display**:
  + Cultural context and historical significance
  + Authentication details and provenance
  + Physical dimensions and condition assessment
* **Categories and Filtering Options**:
  + Region-based classification
  + Material-based grouping
  + Era/period categorization
  + Searchable parameters

# **2.3 Bidding System:**

The bidding mechanism facilitates transparent and fair auction processes:

* Bid Submission Interface:
  + User-friendly bid entry form
  + Clear display of current highest bid
  + Minimum bid increment enforcement
  + Bid History Tracking and Visualization:
  + Chronological bid history for each artifact
  + Visual representation of bidding patterns
  + User-specific bid tracking
* Automatic Minimum Bid Calculation:
  + Automatic calculation updates the minimum bid from the present top offer
  + Percentage-based or fixed increment options
  + A validation system prevents questionable bids at predefined amounts.
* Bid Status Tracking:
  + These states correspond to pending, accepted, outbid and rejected.
  + Real-time status updates
  + Notification system for status changes
* Bid Activity Logging:
  + An extensive record maintains each bid procedure action.
  + Audit trail for system integrity
  + Separate log file implementation

# **2.4 Database Integration:**

All system information requires reliable storage through the data persistence layer which functions to retrieve and save data effectively.

* Entity Framework Core with SQL Server:
  + Object-relational mapping for intuitive data access
  + Strong typing for model validation
  + Transaction support for data integrity
* Code-First Migrations (Microsoft, 2024):
  + Versioned database schema evolution
  + Automated migration generation and application
  + Rollback capability for development flexibility
* Seeding for Initial Data:
  + Default user accounts for testing
  + Sample artifacts with complete details
  + Predefined categories and regions
* Data Relationships:
  + User-to-Artifact (one-to-many)
  + Artifact-to-Bid (one-to-many)
  + User-to-Bid (one-to-many)
  + Category hierarchies (parent-child relationships)

**2.5 African-Themed UI Design:**

The interface embodies African cultural design styles through modern usability guidelines.

* Cultural Color Palette:
  + Earth tones (terracotta, ochre, umber)
  + Nightsea applies traditional textile-inspired colors as accent hues to its interface.
  + Consistent application across all interface elements
* African-Inspired Visual Elements:
  + Pattern motifs from various regional traditions
  + Typography reflecting cultural heritage
  + Iconography based on traditional symbols
* Responsive Layout for All Devices (Bootstrap, 2024):
  + Mobile-first design approach
  + Fluid grid system for different screen sizes
  + Touch-friendly interface elements
  + Consistent experience across platforms

**3. Technology Stack:**

This application adopts contemporary technologies to provide both performance and long-term support through its solution:

* Backend: ASP.NET Core 8.0 MVC
  + C# programming language
  + Razor view engine
  + Dependency injection framework
  + Middleware pipeline architecture
* Database: SQL Server (localdb)
  + Relational data storage
  + Transaction support
  + Data integrity constraints
  + Efficient indexing
* ORM: Entity Framework Core
  + Code-first approach
  + LINQ query capabilities
  + Change tracking
  + Migration support
* Authentication: ASP.NET Core Identity
  + Secure password hashing
  + Role-based authorization
  + Account management
  + External authentication providers support
* Frontend: HTML, CSS, JavaScript, Bootstrap 5
  + Semantic HTML5 markup
  + CSS3 with custom African-themed styles
  + Modern JavaScript (ES6+)
  + Bootstrap 5 for responsive layouts
* Other Libraries:
  + Font Awesome for iconography
  + jQuery Validation for client-side form validation
  + Select2 for enhanced dropdown controls
  + Moment.js for date/time handling

## 

# **4. Project Structure:**

AfriBase/

├── Controllers/ # Controller classes

├── Models/ # Data models and ViewModels

├── Views/ # Razor views organized by controller

├── Data/ # Database context and migrations

├── Services/ # Application services

├── wwwroot/ # Static files (CSS, JS, images)

└── Program.cs # Application entry point

# **5. Setup Instructions:**

## Prerequisites:

* Visual Studio 2022 or later
* .NET 8.0 SDK
* SQL Server LocalDB

## **Steps to Run the Application:**

Clone the Repository:

git clone [repository-url]

cd AfriBase

Open the Solution:  
 Open AfriBase.sln in Visual Studio.

Restore NuGet Packages:  
 Right-click the solution → Restore NuGet Packages.

Update Database Connection:  
 Verify the connection string exists in appsettings.json.

Apply Migrations:  
 Open Package Manager Console and run:

Update-Database

Run the Application:  
 Press F5 or click "Run" in Visual Studio.

Access the Application:  
 Navigate to:<https://localhost:5001>

## **Default Accounts(Used For Testing before DataBase):**

| Role | Email | Password |
| --- | --- | --- |
| Admin | admin@afribase.com | Admin@123 |
| User | user@afribase.com | User@123 |

**6. Key Files and Components:**

Models:

ApplicationUser.cs: Extends IdentityUser with additional user properties and relationships.

Artifact.cs: Defines the artifact entity with all necessary properties.

The Bid.cs file in the database contains artifact records which link to both users and artifacts entities.

View Models:

LoginViewModel.cs: For login form with validation.

RegisterViewModel.cs: For user registration with all required fields.

ArtifactViewModel.cs provides the functionality to show artifact details combined with bidding information.

BidViewModel.cs: For handling bid submission and validation.

Controllers:

AccountController.cs: Manages all authentication operations.

ArtifactController.cs: Provides CRUD operations for artifacts.

BidController.cs: Processes bid submissions and validation.

HomeController.cs: Manages landing page and general navigation.

Services:

IBidLogService.cs functions as a service to write bid logging records into an external log file.

IBidLogService:

Data:

The ApplicationDbContext.cs file contains the EF Core database context while defining all relationships inside it.

The DbInitializer.cs file contains seed functions that create default users and sample artifacts for the system.

# **7. Features Implementation Details:**

User Authentication:

ASP.NET Core Identity manages the authentication alongside authorization system according to Microsoft (2024). The system implements secure user management by using password hashing mechanisms and includes both role-based authorization and account recovery tools. Each user under the system obtains access to system features according to their designated roles while regular users have distinct permission levels from administrators.

Artifact Management:

Through administrator access users can perform Create Read Update and Delete functions to handle content in the artifact administration system. The system presents artifacts by showing their original name and description along with origin information and condition statements and present image data. Multiple conditions can trigger the intelligent sorting functionality which enables filtering in the system.

Bidding System:

A system function automates the bidding process by validating that each entered amount satisfies the minimum criterion. Every recorded bidding history and its corresponding visual data is easily retrievable by users. Concurrent bidding operates through the system with locking procedures developed for data protection.

African-Themed Design:

The interface builds authenticity by combining earth-toned colors with African patterns as well as African-inspired typographical elements (Font Awesome, 2024; Bootstrap, 2024). The system features African-inspired CSS variables together with SVG pattern backgrounds alongside African design-inspired button design elements.

# **8. Future Enhancements:**

Email Notifications:

The website needs to use actual email services to restore forgotten passwords and notify users about bid status changes.

Users should have an option to adjust the reception settings of their notifications.

The system includes HTML email templates featuring African design elements.

Advanced Search:

The application needs advanced features to search and filter artifact data sets.

The application should implement search functions that use multiple filter criteria.

The platform should offer regional artifact search through specific location items.

Payment Integration:

Customers should have a built-in payment processing gateway through which they can finalize successful winning bids.

Support multiple currency options.

High-value transactions should include escrow services deployed for protection purposes.

Artifact Collections:

Group artifacts by regional or thematic collections.

Organize specially selected exhibitions which provide historical frameworks for artifacts.

Enable collection sharing and embedding.

Multilingual Support:

The system requires an implementation of translation capabilities specifically for African language groups.

The system needs to display monetary values and dates according to local conventions.

The system should allow users to select from appropriate regional dialects.

## **9. Conclusion:**

The AfriBase development team succeeded in creating a robust web application that implements ASP.NET Core MVC development guidelines. The web application merges all elements including responsive African-themed design as well as user authentication security and data storage features and user administration mechanism. Integrated cultural elements enable the application to create an interactive auction system through its structured technology platform (Sharma, 2020).

All requirements from the project brief can be found with success in the application which features secure authentication alongside artifact control and bidding mechanisms and database synchronization alongside African-style design elements. Modern technology implementation together with best practices enables the system to be secure and sustainable as well as expandable which forms a base for upcoming developments. (Sharma, 2020)

## 

## **10. Flowchart:**

# 

## 

## **Page Design:**

## **Register(working fully):**

## 

## **Login(Working fully):**

## 

## **HomePage Design(After Signing in or Registering):**

## 

## 

## 

## 

## 

## 

## 

## **ArtifactPage(When logged in username will show):**

## 

## 

## **BidDashboard(Administrator Login):**

## 

## **RegionsPage:**

## 

## 

## **CategoriesPage:**

## 

## **ExploreCategory:**

## 

## **User ProfilePage(after database was created)**

## **User BidHistory(accessed from profile)**

## **12. References:**

* Microsoft. (2024). ASP.NET Core MVC Overview. Retrieved from https://learn.microsoft.com/en-us/aspnet/core/mvc/overview
* Microsoft. (2024). ASP.NET Core Identity Overview. Retrieved from https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity
* Microsoft. (2024). Entity Framework Core Documentation. Retrieved from https://learn.microsoft.com/en-us/ef/core/
* Bootstrap. (2024). Bootstrap Documentation (v5.3). Retrieved from https://getbootstrap.com/docs/5.3/getting-started/introduction/
* Font Awesome. (2024). Font Awesome Icons Documentation. Retrieved from https://fontawesome.com/docs/
* Sharma, S. (2020). ASP.NET Core: A Server-Side Web Application Framework. LinkedIn. Retrieved from https://www.linkedin.com/pulse/aspnet-core-server-side-web-application-framework-sunil-sharma/
* jQuery Foundation. (2024). jQuery Validation Plugin Documentation. Retrieved from https://jqueryvalidation.org/documentation/