VIRTUAL ZOO

Immersive 3D Ecosystem Exploration Platform

Project Report

Date: November 2025

Table of Contents

1. INTRODUCTION

1.1 PURPOSE

This document provides a comprehensive description of the Virtual Zoo web application, an immersive educational platform designed for exploring ecosystems, learning about animals, and participating in educational sessions. The purpose of this report is to document the system's requirements, design, implementation, and functionality.

The Virtual Zoo platform serves as an educational tool that allows students, teachers, and administrators to interact with virtual ecosystems, view detailed information about various species, and participate in structured learning sessions.

1.2 SCOPE

The Virtual Zoo application encompasses the following scope:

• **Ecosystem Management:** Creation, viewing, and management of various ecosystems (Amazon Rainforest, Sahara Desert, Arctic Tundra, etc.)

• **Animal Database:** Comprehensive database of existing and extinct species with detailed information

• **Educational Sessions:** Video-based learning sessions with quizzes, comments, and downloadable resources

• **User Management:** Role-based access control for Admin, Teacher, and Student users

• **Progress Tracking:** Student progress tracking for visited ecosystems and attended sessions

• **Content Management:** Admin and teacher panels for content creation and management

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Term	Definition
Django	High-level Python web framework
PostgreSQL/SQLite	Database management systems
TailwindCSS	Utility-first CSS framework
CRUD	Create, Read, Update, Delete operations
ORM	Object-Relational Mapping
MVC	Model-View-Controller architecture pattern
REST	Representational State Transfer
API	Application Programming Interface
URL	Uniform Resource Locator
HTML	HyperText Markup Language

CSS	Cascading Style Sheets
JS	JavaScript

1.4 OVERVIEW

This document is organized into seven main sections:

• **Introduction:** Purpose, scope, and overview

• **General Description:** Product perspective, functions, user characteristics

• **Specific Requirements:** Interface requirements, functional and non-functional requirements

• **Website Screenshots:** Visual documentation of the application

• **Code Screenshots:** Key implementation details

• **Database Schema:** Database structure and relationships

• **Project Structure:** Deployment and repository information

2. GENERAL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

The Virtual Zoo is a standalone web application built using Django framework. It operates as an educational platform that can be deployed on any web server supporting Python and Django. The system integrates with PostgreSQL (production) or SQLite (development) databases, TailwindCSS for styling, and local file system for media storage.

2.2 PRODUCT FUNCTIONS

The Virtual Zoo platform provides the following core functions:

Ecosystem Explorer: Browse ecosystems by region and era, filter and search, view detailed environmental information

Animal Database: View comprehensive animal information, filter by species type, search functionality

Educational Sessions: Video-based learning, session enrollment, downloadable resources, interactive quizzes

User Management: Role-based authentication, user registration, profile management, progress tracking

Content Management: Admin panel for ecosystem/animal creation, teacher panel for session management

2.3 USER CHARACTERISTICS

The system serves three distinct user roles:

Administrators: Full system access, can create/edit/delete ecosystems and animals, user management capabilities

Teachers: Create and manage educational sessions, upload resources, create quizzes, view student enrollments

Students: Browse ecosystems and animals, enroll in sessions, track learning progress, access resources

2.4 GENERAL CONSTRAINTS

• Browser Compatibility: Modern browsers (Chrome, Firefox, Safari, Edge)

• Network: Requires internet connection for video playback

• Storage: Local file storage for uploaded media

• Performance: Optimized for desktop, tablet, and mobile devices

• Security: Django's built-in security features (CSRF protection, SQL injection prevention)

2.5 ASSUMPTIONS AND DEPENDENCIES

Assumptions:

• Users have basic web browsing knowledge

• Video content hosted on external platforms (YouTube/Vimeo)

• Images are provided or downloaded from external sources

• Database server is accessible and properly configured

Dependencies:

• Python 3.8+

• Django 4.2+

• PostgreSQL 12+ (production) or SQLite (development)

• Node.js and npm (for TailwindCSS compilation)

• Web server (for production deployment)

3. SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 User Interfaces

The application provides a modern, responsive web interface with homepage hero section, navigation bar, ecosystem list with grid layout, ecosystem detail pages, session detail pages with video players, user dashboards, and TailwindCSS-styled forms for all CRUD operations.

3.1.2 Hardware Interfaces

Server Requirements: Minimum 2GB RAM, 10GB storage space, network connectivity. Client Requirements: Modern web browser, internet connection, JavaScript enabled.

3.1.3 Software Interfaces

Backend: Django 4.2+ framework, Python 3.8+ runtime, Database: PostgreSQL or SQLite. Frontend: TailwindCSS 3.0+, HTML5, JavaScript (vanilla). External Services: YouTube/Vimeo API for video embedding.

3.1.4 Communications Interfaces

HTTP/HTTPS: Standard web protocols. RESTful URLs: Clean URL structure.

3.2 FUNCTIONAL REQUIREMENTS

FR1: User Authentication

• Users can register with username, email, password, and role

• Users can log in with credentials

• Users can log out securely

• Password validation and security

FR2: Ecosystem Management

• View list of all ecosystems with pagination

• Filter ecosystems by region, era, and search query

• View detailed ecosystem information

• Admin/Teacher can create, edit, and delete ecosystems

• Upload and display ecosystem images

FR3: Animal Management

• View animals associated with ecosystems

• Filter animals by species type (existing/extinct)

• View detailed animal information

• Admin/Teacher can create, edit, and delete animals

• Display animal images and conservation status

FR4: Educational Sessions

• View list of available sessions

• View session details with embedded video player

• Students can enroll/unenroll in sessions

• Teachers can create, edit, and delete sessions

• Upload and download session resources

• Interactive quizzes with multiple choice questions

• Comment system for session discussions

FR5: Progress Tracking

• Track student visits to ecosystems

• Track student attendance in sessions

• Display progress statistics in student dashboard

• Time spent tracking for learning analytics

3.3 NON-FUNCTIONAL REQUIREMENT

NFR: Performance

Page load time: < 2 seconds, Database queries optimized, Image optimization, Pagination for large datasets

NFR: Usability

Intuitive navigation, Responsive design, Clear visual hierarchy, Accessible color schemes, User-friendly error messages

NFR: Security

CSRF protection enabled, SQL injection prevention, XSS protection, Secure password hashing, Role-based access control

NFR: Reliability

Error handling and logging, Database transaction management, Graceful degradation, Backup and recovery procedures

NFR: Maintainability

Clean code structure, Comprehensive documentation, Modular design, Follows Django best practices

NFR: Scalability

Database indexing, Efficient query optimization, Support for multiple concurrent users, Media file organization

3.4 DESIGN CONSTRAINTS

Technology Constraints:

Must use Django framework, Python 3.8+ required, PostgreSQL or SQLite database, TailwindCSS for styling

Platform Constraints:

Web-based application (no native mobile apps), Requires modern browser support, JavaScript must be enabled

3.5 DJANGO AND DATABASE FEATURES

Django Framework Features:

• ORM (Object-Relational Mapping): Database abstraction layer

• Admin Interface: Built-in content management

• Template System: Django template language

• Form Handling: Django forms with validation

• Authentication: Built-in user authentication system

• Middleware: Security and session management

• URL Routing: Clean URL patterns

• Migrations: Database schema version control

Database Models Implemented:

• User Model (Custom): Role-based user system (Admin, Teacher, Student)

• Ecosystem Model: Name, description, location, region, era, environmental data, image upload

• Animal Model: Name, scientific name, species type, conservation status, habitat, diet, image upload

• EducationalSession Model: Title, description, session type, teacher, scheduled date, video URL, lesson content

• SessionResource Model: File uploads (PDFs, documents), title and description

• SessionComment Model: User comments on sessions with timestamp tracking

• SessionQuiz Model: Multiple choice questions with options and correct answers

- SessionEnrollment Model: Student enrollment tracking with attendance status
- StudentProgress Model: Tracks ecosystem visits, session attendance, time spent tracking

4. SCREENSHOTS OF WEBSITE

4.1 HOME PAGE

The homepage features a modern hero section with gradient background (emerald to teal to cyan), animated floating animal icons, and three prominent purple action buttons. Below the hero section are featured ecosystems and upcoming educational sessions displayed in card layouts.

Note: Screenshots should be captured showing the full hero section with gradient background, featured ecosystems grid, upcoming sessions display, and responsive design on different screen sizes.

4.2 ECOSYSTEM EXPLORER

The ecosystem explorer provides a comprehensive interface for browsing and filtering ecosystems. Features include filter by region and era, search functionality, grid layout with ecosystem cards. Detail pages show large ecosystem image header, environmental information, and associated animals list.

4.3 LOGIN/REGISTRATION PAGE

Clean, user-friendly authentication pages with TailwindCSS styling. Login page includes username and password fields, 'Remember me' option, and link to registration. Registration page includes username, email, password fields, role selection, and additional fields for first name, last name, phone, and bio.

4.4 EDUCATIONAL SESSIONS

Comprehensive session management and viewing interface. Session list page shows grid layout of session cards with session type badges, teacher information, enrollment status, and scheduled dates. Session detail page includes video player (YouTube/Vimeo embed), session description, enrollment button, downloadable resources section, quiz section, comments section, and lesson content display.

4.5 USER DASHBOARDS

Student Dashboard: Visited ecosystems count, enrolled sessions count, learning time statistics, list of visited ecosystems, list of enrolled sessions, progress tracking.

Teacher Dashboard: Created sessions list, enrollment statistics, session management options, resource management, student engagement metrics.

5. SCREENSHOTS OF CODE

5.1 Model Definitions

File: ecosystem/models.py

```
class Ecosystem(models.Model):
    REGION_CHOICES = [
        ('amazon', 'Amazon Rainforest'),
        ('sahara', 'Sahara Desert'),
        ('arctic', 'Arctic Tundra'),
    ]

    name = models.CharField(max_length=200)
    description = models.TextField()
    location = models.CharField(max_length=200)
    region = models.CharField(max_length=50, choices=REGION_CHOICES)
    era = models.CharField(max_length=50, choices=ERA_CHOICES)
    temperature_min = models.DecimalField(max_digits=5, decimal_places=2)
    temperature_max = models.DecimalField(max_digits=5, decimal_places=2)
    vegetation = models.TextField(blank=True)
    precipitation = models.CharField(max_length=100, blank=True)
    image = models.ImageField(upload_to='ecosystems/', blank=True, null=True)
    created_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

5.2 View Functions

File: ecosystem/views.py

```
def ecosystem_list(request):
    ecosystems = Ecosystem.objects.all()

    # Filtering logic
    region_filter = request.GET.get('region')
    era_filter = request.GET.get('era')
    search_query = request.GET.get('search')

    if region_filter:
        ecosystems = ecosystems.filter(region=region_filter)
    if era_filter:
        ecosystems = ecosystems.filter(era=era_filter)
    if search_query:
        ecosystems = ecosystems.filter(
            Q(name__icontains=search_query) |
            Q(description__icontains=search_query)
        )

    paginator = Paginator(ecosystems, 9)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    return render(request, 'ecosystem/list.html', {
        'page_obj': page_obj,
        'region_filter': region_filter,
        'era_filter': era_filter,
        'search_query': search_query,
    })
```

6. SCREENSHOTS OF DATABASE SCHEMA

6.1 Entity Relationship Diagram

Main Entities: User (Admin, Teacher, Student), Ecosystem, Animal, EducationalSession, SessionResource, SessionComment, SessionQuiz, SessionEnrollment, StudentProgress

Relationships:

• User 1:N Ecosystem (created_by)

• Ecosystem 1:N Animal

• User 1:N EducationalSession (teacher)

• Ecosystem 1:N EducationalSession (optional)

• EducationalSession 1:N SessionResource

• EducationalSession 1:N SessionComment

• EducationalSession 1:N SessionQuiz

• EducationalSession 1:N SessionEnrollment

• User 1:N StudentProgress

6.2 Database Tables

Table Name	Columns
accounts_user	id, username, email, password, role, phone_number, bio, created_at, updated_at
ecosystem_ecosystem	id, name, description, location, region, era, climate, temperature_min, temperature_max, veget
ecosystem_animal	id, ecosystem_id, name, scientific_name, species_type, description, habitat, diet, conservation_
educational_sessions_educationalsession	id, title, description, session_type, teacher_id, ecosystem_id, scheduled_date, duration_minute
educational_sessions_sessionresource	id, session_id, title, file, description, uploaded_at
educational_sessions_sessioncomment	id, session_id, user_id, content, created_at, updated_at
educational_sessions_sessionquiz	id, session_id, question, option_a, option_b, option_c, option_d, correct_answer, explanation, c
educational_sessions_sessionenrollment	id, session_id, student_id, enrolled_at, attended, notes
accounts_studentprogress	id, student_id, ecosystem_id, session_id, visited_at, time_spent_minutes, completed

7. PROJECT STRUCTURE AND DEPLOYMENT

7.1 Project Structure

```
virtual_zoo/
■■■ accounts/              # User management app
■   ■■■ models.py          # User and StudentProgress models
■   ■■■ views.py           # Authentication and dashboard views
■   ■■■ forms.py           # Registration form
■   ■■■ templates/         # User-related templates
■■■ ecosystem/            # Ecosystem management app
■   ■■■ models.py          # Ecosystem and Animal models
■   ■■■ views.py           # CRUD operations for ecosystems
■   ■■■ forms.py           # Ecosystem and Animal forms
■   ■■■ templates/         # Ecosystem templates
■   ■■■ management/        # Management commands
■■■ educational_sessions/ # Educational sessions app
■   ■■■ models.py          # Session, Resource, Comment, Quiz models
■   ■■■ views.py           # Session management views
■   ■■■ forms.py           # Session forms
■   ■■■ templates/         # Session templates
■■■ virtual_zoo/           # Main project directory
■   ■■■ settings.py        # Django settings
■   ■■■ urls.py            # Main URL configuration
■   ■■■ templates/         # Base templates
■■■ theme/                 # TailwindCSS theme
■■■ media/                 # Uploaded media files
■■■ static/                # Static files
■■■ manage.py             # Django management script
```

7.2 Installation and Setup

Requirements: Python 3.8+, Django 4.2+, PostgreSQL 12+ (production) or SQLite (development), Node.js and npm (for TailwindCSS)

Installation Steps:

1. Create virtual environment: python -m venv myenv

2. Install dependencies: pip install -r requirements.txt

3. Run migrations: python manage.py migrate

4. Create superuser: python manage.py createsuperuser

5. Setup TailwindCSS: python manage.py tailwind install

6. Create demo data: python manage.py create_demo_data

7. Run development server: python manage.py runserver

7.3 Deployment Considerations

Production Database: Use PostgreSQL instead of SQLite. Static Files: Collect and serve through web server or CDN. Media Files: Store in cloud storage (AWS S3, etc.) for production. Security: Set DEBUG=False, configure ALLOWED_HOSTS. Web Server: Use Gunicorn with Nginx. HTTPS: Enable SSL certificates. Backup: Regular database backups.

7.4 GitHub Repository

Repository Structure: Main branch for production-ready code, Development branch for feature development, README.md with setup instructions, requirements.txt for Python dependencies, .gitignore for excluded files.

Note: Include GitHub repository link in the actual PDF report.