



Interactive Educational Platform for Wildlife Awareness
System Documentation by Zitong Wang

Contents

1	Project Introduction	1
1.1	Problem Statement	1
1.2	Scope	1
1.3	Development Planning	2
2	System Design and Technical Implementation	3
2.1	System Overview and Architecture	3
2.1.1	System Architecture Description	3
2.1.2	Technology Stack Overview	4
2.2	System Functional Design	5
2.2.1	User Roles and Use Cases	5
2.2.2	Functional Module Design and Implementation	6
2.2.3	User Interaction and Event Flow	8
2.2.4	System Workflow and Data Processing	10
2.3	Data Model Design	10
2.3.1	Data Model Overview	10
2.3.2	Entity-Relationship Diagram	11
2.3.3	Data Synchronization and Consistency Strategies	12
2.4	Module Interaction and Communication Design	13

3 System Testing and Quality Assurance 14
3.1 Testing Strategy and Implementation 14
3.2 Security Architecture 14
3.3 UX/UI Design Principles 15
4 Conclusion 15
4.1 Future Prospects 15
4.2 Achievements and Reflections 15

Abstract

Dwen Dwen’s Neighbour is a web-based educational and public welfare platform designed by the QAA Software Engineering Company in response to the urgent need for endangered species protection in China. Commissioned by Chinese Animal Guardians, a non-profit animal protection organization, the platform integrates knowledge dissemination, interactive games, virtual red panda adoption, community engagement, and donation functions into a cohesive digital experience.

The website features five main modules: an encyclopedia of endangered animals with interactive maps and scientifically sourced data; a suite of educational games to promote learning through play; a virtual red panda adoption module powered by Live2D technology; a community forum supporting posts, messaging, and social interaction; and a donation platform with reward incentives. Developed over twelve weeks by a team of six using the Agile methodology, the system employs a monolithic architecture built on Flask, with Firebase Realtime Database and Authentication for real-time data handling and secure access control.

The project emphasizes user-centered design, gamification, and real-time interactivity. It leverages open-source tools and cloud services to ensure scalability and performance. This documentation details the system’s requirements, design, implementation, testing, and team collaboration, serves as a technical reference. Through this project, the team demonstrates the practical application of software engineering principles in building digital systems.

1 Project Introduction

1.1 Problem Statement

With the continuous deterioration of the global ecological environment, many rare animals are facing a growing survival crisis. As a country rich in biodiversity, China is home to numerous unique and endangered species. However, due to habitat destruction, climate change, and the intensification of human activities, the natural habitats of these animals are becoming increasingly fragile.

In response to these pressing challenges, **Chinese Animal Guardians**, a non-profit organization dedicated to the protection of China's wild species and ecosystems, has commissioned **QAQ Software Engineering Company** to design and develop a bespoke website that addresses three core objectives:

- **Promote animal protection knowledge**
- **Enhance public awareness** of local wildlife conservation in China
- **Mobilize the public** to raise funds for the organization's public welfare initiatives

To fulfill these goals, our team has structured the website around five central modules:

1. **Encyclopedia:** A richly illustrated and searchable database of China's at-risk species, complete with distribution maps and conservation facts to educate users effectively.
2. **Fun Games:** A collection of five interactive quizzes (*Guess Who*, *Matching*, *Animal Puzzle*, *Concentration*, and *Challenge*) designed to reinforce learning through engaging and entertaining formats.
3. **My Red Panda:** A virtual red panda raising experience powered by Live2D technology allows users to adopt a virtual red panda, giving it a name, feeding it, playing with it, writing a diary, and taking it on travel. Most of the activities in this section require users to redeem scores earned from the Fun Games, motivating users to actively learn about animals. This feature helps users build an emotional connection while gaining in-depth knowledge about red pandas, encouraging them to understand and join the cause of animal protection.
4. **Community:** A user-driven community within the website where users can post and manage their posts, chat with others, and add friends. This feature aims to increase user engagement by encouraging active sharing of animal-related posts, discussions on wildlife news, and a stronger focus on wildlife conservation.
5. **Donation:** Provide a direct donation portal to Chinese Animal Guardians and open a platform for individuals to freely create their own charitable fundraising campaigns. A rich reward system—including certificates, virtual badges, awards, and more—will be implemented to encourage users to engage in transparent giving.

To create a warm, recognizable, and friendly identity, the website is named **“Dwen Dwen's Neighbour”**. This name draws inspiration from the beloved 2022 Winter Olympics mascot *“Bing Dwen Dwen”*, symbolizing friendliness, cuteness, and Chinese cultural identity. The name reflects the platform's mission: to serve as a welcoming space focused on Chinese wildlife and accessible to global audiences.

Ultimately, this platform will empower **Chinese Animal Guardians** to disseminate wildlife knowledge, galvanize public support, and promote sustainable fundraising—all within a cohesive, engaging, and user-friendly digital environment.

1.2 Scope

Dwen Dwen's Neighbour supports two types of users: **regular users** and **administrators**. While both user groups have access to the website's core functionalities, administrators are granted additional privileges. These include access to the Firebase console, where they can use Firebase Authentication to manage user accounts — such as viewing account details, banning users, and deleting accounts when necessary.

This system documentation is designed to provide both user types with clear, structured, and comprehensive guidance on using the website effectively. It also serves as a technical reference for stakeholders of the QAQ company and partner environmental organizations. The document facilitates consistent system usage and maintenance, supporting broader efforts to protect endangered animals in China.

1.3 Development Planning

The project was completed over a period of twelve weeks by a team of six members, following the Agile methodology. The planning phase involved requirements analysis, database schema design, iterative development using HTML, CSS, JavaScript, and Flask, UI/UX design, as well as testing and deployment. Our team conducted regular meetings and utilized task tracking tools to ensure efficient collaboration and timely delivery.

Below is a detailed task list for the project, outlining each subtask and its associated deadline for every feature (see Figure 1).

	Welcome Page	Registration /login	Information Management	Map Display	Animal Information	Sounds & Details	Classification & Search	Donation	Medals & Rewards	Create Activities	Certificate & Records	
Database		3.12 ✔	3.13 ✔		3.16 ✔	3.16 ✔		4.7 ✔	4.7 ✔	4.8 ✔	4.9 ✔	
Backend	3.12 ✔	3.12 ✔	3.13 ✔	3.20 ✔	3.21 ✔	3.21 ✔	3.22 ✔	4.7 ✔	4.8 ✔	4.8 ✔	4.9 ✔	
Frontend	3.12 ✔	3.12 ✔	3.13 ✔	3.20 ✔	3.21 ✔	3.21 ✔	3.22 ✔	4.7 ✔	4.8 ✔	4.8 ✔	4.9 ✔	
UI	4.4 ✔	4.1 ✔	4.1 ✔	4.3 ✔	4.3 ✔	4.3 ✔	4.3 ✔	4.10 ✔	4.10 ✔	4.10 ✔	4.10 ✔	
Deployment	4.6 ✔	4.6 ✔	4.6 ✔	4.6 ✔	4.6 ✔	4.6 ✔	4.6 ✔	4.10 ✔	4.10 ✔	4.10 ✔	4.10 ✔	
Testing	4.6 ✔	4.6 ✔	4.6 ✔	4.6 ✔	4.6 ✔	4.6 ✔	4.6 ✔	4.10 ✔	4.10 ✔	4.10 ✔	4.10 ✔	
	2D Live	Certificate	Feed & Play	Diary	Travel	Post System	Comments & likes	Tagging & Search	Add friends & chat	Menu	Game Logic	Score Statistics
Database		3.18 ✔	3.19 ✔	3.28 ✔	4.27 ✔	4.1 ✔	4.1 ✔	4.16 ✔	4.5 ✔			3.19 ✔
Backend	3.21 ✔	3.18 ✔	3.18 ✔	3.31 ✔	4.27 ✔	4.7 ✔	4.8 ✔	4.16 ✔	4.16 ✔	3.19 ✔	3.23 ✔	3.20 ✔
Frontend	3.21 ✔	3.18 ✔	3.18 ✔	3.31 ✔	4.27 ✔	4.7 ✔	4.8 ✔	4.16 ✔	4.16 ✔	3.19 ✔	3.23 ✔	3.20 ✔
UI	4.30 ✔	4.30 ✔	4.30 ✔	4.30 ✔	4.30 ✔	4.17 ✔	4.17 ✔	4.17 ✔	4.17 ✔	4.2 ✔	4.2 ✔	4.2 ✔
Deployment	3.26 ✔	5.2 ✔	5.2 ✔	5.2 ✔	5.2 ✔	4.10 ✔	4.10 ✔	4.20 ✔	4.20 ✔	4.6 ✔	4.6 ✔	4.6 ✔
Testing	3.26 ✔	5.4 ✔	5.4 ✔	5.4 ✔	5.4 ✔	4.10 ✔	4.10 ✔	4.20 ✔	4.20 ✔	4.6 ✔	4.6 ✔	4.6 ✔

Figure 1: Project Task List Overview

Additionally, the technologies used throughout the project are summarized in the technology list (see Figure 2). This list highlights the main frameworks, libraries, and tools employed to implement the system, ensuring a robust and scalable solution.

	User System	Encyclopedia	Games	My Red Panda	Community	Donation
Firestore Realtime Database	✓	✓	✓	✓	✓	✓
Flask	✓	✓	✓	✓	✓	✓
Firestore Admin SDK	✓	✓	✓	✓	✓	✓
Firestore Authentication	✓	✗	✗	✗	✗	✗
APScheduler	✓	✓	✓	✓	✓	✓
Requests & BeautifulSoup	✗	✓	✗	✗	✗	✗
CORS	✓	✓	✓	✓	✓	✓
HTML/CSS	✓	✓	✓	✓	✓	✓
Tailwind CSS	✓	✓	✓	✓	✓	✓
JavaScript	✓	✓	✓	✓	✓	✓
ECharts	✗	✓	✗	✓	✗	✗
Font Awesome & Google Fonts	✓	✓	✓	✓	✓	✓
Layered SVG interactive animation	✗	✗	✗	✓	✗	✗
Bootstrap	✗	✓	✓	✓	✗	✗
Swiper JS	✗	✗	✗	✗	✓	✗

Figure 2: Technology Stack and Tools Used

2 System Design and Technical Implementation

2.1 System Overview and Architecture

2.1.1 System Architecture Description

Dwen Dwen's Neighbour is developed as a monolithic web application based on the Flask framework, integrating both frontend and backend logic. The system adopts the MVC (Model-View-Controller) design pattern and uses Jinja2 for server-side HTML rendering. Static assets and dynamic functionalities are managed uniformly within this architecture. The system architecture consists of the following layers:

- **Application Layer:** Handles HTTP requests, routing, and business logic using Flask.
- **Presentation Layer:** Uses Jinja2, HTML, CSS, and JavaScript to render and manage dynamic content.
- **Data Layer:** Utilizes Firebase Realtime Database for cloud-based data storage and real-time synchronization.
- **Authentication Layer:** Implements Firebase Authentication to provide a robust and secure mechanism for user login, registration, password management, and session control.
- **Auxiliary Modules:** Includes web scraping tools (Requests + BeautifulSoup) for dynamically retrieving species information, and ECharts for interactive data visualizations.

The implementation of this architectural design is illustrated in Figure 3.

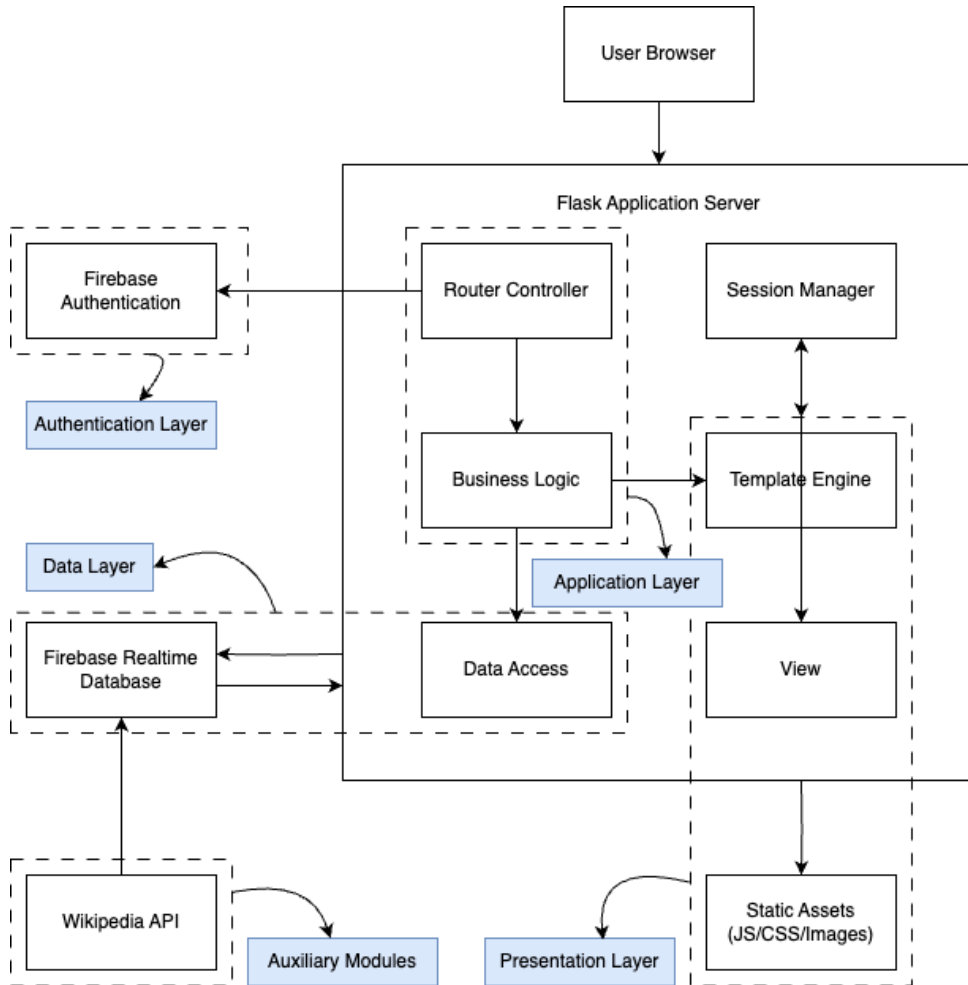


Figure 3: System Architecture Diagram

2.1.2 Technology Stack Overview

Table 1 summarizes the core technologies used in the system, organized by their functional roles:





Category	Technologies and Usage
Backend 	<ul style="list-style-type: none"> • Flask: Lightweight Python framework for HTTP handling • Flask-CORS: Enables cross-origin communication • Flask-Session: Manages server-side sessions • APScheduler: Schedules background data updates • Firebase Admin SDK: Access to Database and Authentication services
Frontend   	<ul style="list-style-type: none"> • HTML5: Markup structure of all pages • CSS3: Styling and responsive layout • JavaScript: Client-side logic and interactions • Jinja2: Server-side HTML template rendering • Bootstrap: Ensures responsive and aesthetic UI • ECharts: Visualization library for interactive geographic data • Live2D (SVG): Red panda character animation • Font Awesome: Vector icon library
Data & Auth 	<ul style="list-style-type: none"> • Firebase Realtime Database: NoSQL cloud database with real-time sync • Firebase Authentication: Secure user auth and session control
External Data  	<ul style="list-style-type: none"> • Requests: Python HTTP client • BeautifulSoup: Extracts animal data from HTML pages (Wikipedia)

Table 1: Overview of Technologies Used in the System

2.2 System Functional Design

2.2.1 User Roles and Use Cases

The system defines two distinct types of users with different access privileges (see Table 2):

User Role	Access and Functionalities
General User	Has full access to all functionalities of the website, including encyclopedia, games, my red panda, community, donation and profile.
Administrator	Inherits all general user permissions. Additionally, administrators have privileged access to the Firebase Console to manage user accounts, including viewing, suspending, and deleting them.

Table 2: Comparison of User Roles

User interactions with system functionalities are shown in Figure 4.

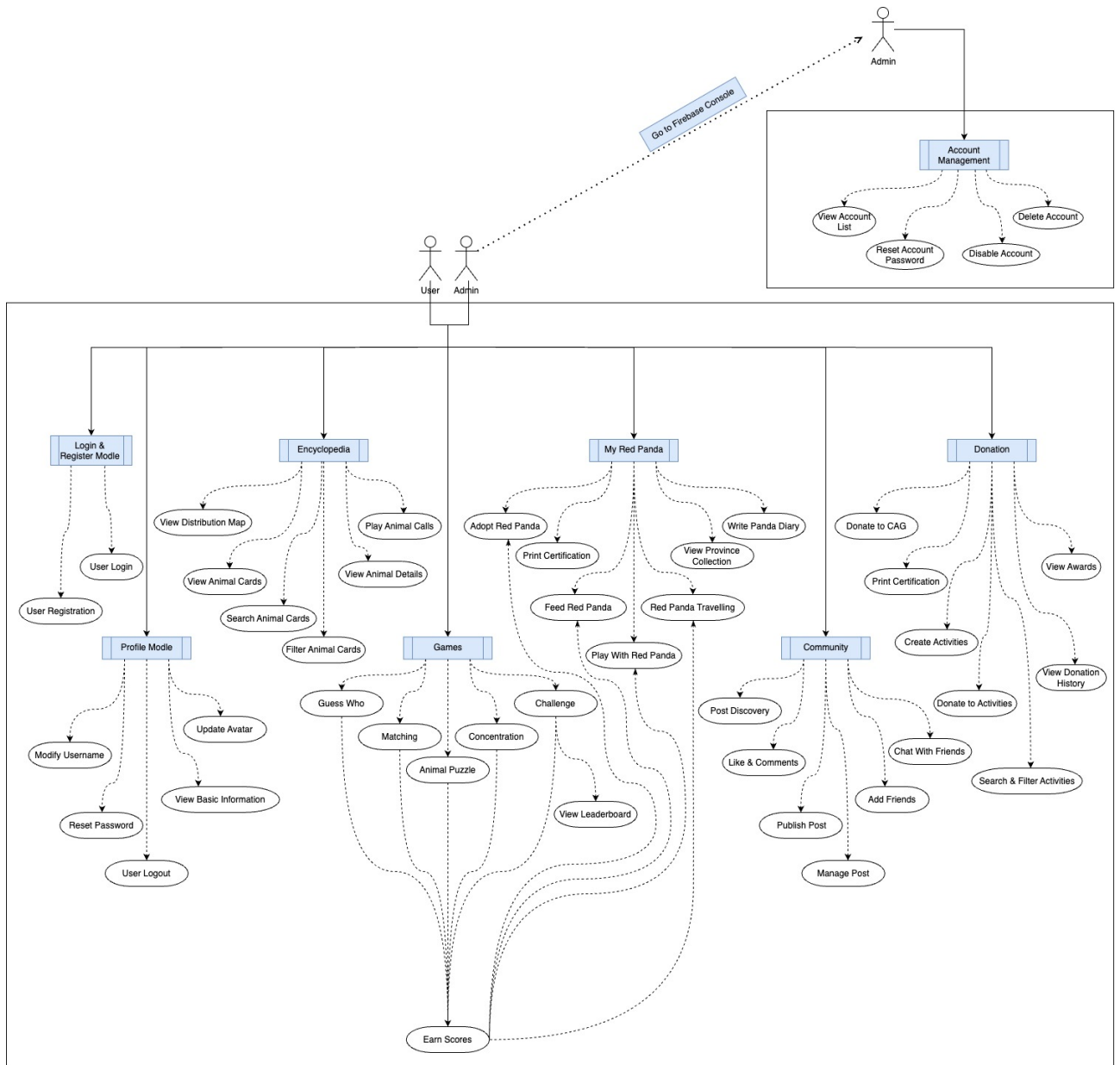


Figure 4: Use Case Diagram

2.2.2 Functional Module Design and Implementation

Account Management Module

- Administrators are granted additional privileges to access the Firebase Console, a separate management platform outside the website, where they can view all user information, disable accounts, or permanently delete accounts. This access helps ensure the security and integrity of the platform.
 1. View and monitor the entire user database in real time
 2. Manage user accounts, including disabling or permanently deleting accounts
 3. Ensure system integrity, detect misuse, and respond to inappropriate behavior

User Management Module

- **User Registration**

The system features a complete registration workflow, leveraging Firebase Authentication for identity verification. The registration form collects the username, email, and password. Upon account creation, extended user information is stored in the Firebase Realtime Database.
- **User Login**

The login system validates user credentials via Firebase Authentication and manages user sessions using Flask session management. Persistent sessions are generated post-login, allowing users to remain authenticated for a fixed duration. Key design aspects include:

 1. Session-based authentication mechanism utilizing `session` for user information storage
 2. Middleware for request interception to automatically verify user login status
 3. AJAX support for asynchronous login requests and real-time status updates
- **User Profile Management**

Users can update personal information, change passwords, and customize avatars on the profile page. The system employs real-time data synchronization to ensure instant effect of account modifications. Key features include:

 1. Avatar upload and cropping functionality
 2. Username modification with synchronization between Firebase Authentication and Realtime Database
 3. Secure password updating with validation
 4. Display of user points, red panda adoption history, and donation records
 5. Logout current account

Encyclopedia Module

- **Geographic Data Visualization**

ECharts is utilized to implement interactive visualizations of the national and provincial maps of China, illustrating wildlife distribution. Users can click on provinces to navigate and explore endemic species in different regions.
- **Animal Encyclopedia Information System**

The system retrieves and updates animal data using web scraping from Wikipedia, including classification, habitats, conservation status, and detailed descriptions. A scheduled background task ensures regular data updates.
- **Animal Cards**

Each card includes animal vocalizations, names in Chinese, English, and Latin (with pronunciation tools), and distribution data, providing users with comprehensive species knowledge.
- **Categorization by Type and Conservation Level**

Animal information is classified according to IUCN Red List categories (e.g., Critically Endangered – CR, Endangered – EN, Vulnerable – VU), facilitating systematic understanding of endangered species protection.

Game Module

- **Game Type Design**

The platform offers five educational game types, each targeting different learning methods and knowledge areas:

1. **Guess Who:** Identifying animals through images to cultivate visual recognition skills
2. **Matching:** Associating animal names with images to enhance associative memory
3. **Animal Puzzle:** Jigsaw puzzles to improve attention to animal morphology
4. **Concentration:** Card-flipping game to train memory
5. **Challenge Mode:** Matching English, Chinese, and scientific names to enhance terminology mastery
This mode also records historical high scores, encouraging self-challenge and featuring a leaderboard to stimulate competition

- **Points Reward Mechanism**

Upon completion of games, users earn points, which are added to their total score. Points can be used for adopting virtual red pandas, purchasing virtual items, and participating in virtual donations.

- **Game Data Management**

Animal data used in games is dynamically retrieved from Firebase Realtime Database, ensuring content freshness and accuracy.

My Red Panda Module

- **Red Panda Adoption Mechanism**

Users can adopt a virtual red panda using scores, name the pet, and initiate the nurturing process. The system records the adoption time and generates an adoption certificate to enhance user engagement.

- **Live Status System**

Red panda has three dynamic attributes - fullness, mood, and fatigue - which automatically evolve over time and require users to maintain its attribute health through regular interactive behavior.

- **Interaction Mechanism**

Users can interact with their red panda through:

1. Feeding: Consumes points to increase satiety, with different effects per food type(-5 scores)
2. Playing: Improves mood while increasing fatigue
3. Traveling: Takes the panda to different provinces in China, logging visited regions(-10 scores)

- **Provinces Collection Feature**

After traveling with the panda, users collect provincial map entries and receive regional friendliness ratings, encouraging exploration across the country.

Community Module

- **Post Publishing**

Users can publish posts containing text, images, and multimedia content to share knowledge and experiences in wildlife conservation. Posts are displayed in real-time on the community page.

- **Interaction System**

Posts support comment functionality, enabling user discussions. A “like” feature is also implemented to promote quality content.

- **Post Management**

Users may manage their posts, including editing and deleting published content.

- **Add Friends**

The system allows users to search and add friends, fostering social connections among users.

- **Chat Functionality**

Friends can engage in one-on-one messaging, with message sending, receiving, and history tracking features.

Donation Module

• Donation Campaigns

Users can donate directly to Chinese Animal Guardians or to user-initiated public welfare projects. The platform permits free creation of charitable projects by users.

• Rewards and Achievement System

Upon donating, users receive virtual rewards such as badges and certificates. The platform incorporates a tiered badge system and special rewards for first-time donors to enhance a sense of achievement.

• Donation Progress Tracking

The system displays real-time fundraising progress for each project. Users can view their personal donation history and individual contributions to various campaigns.

2.2.3 User Interaction and Event Flow

Event: Browsing Animal List by Clicking on the Distribution Map

User Story: The user views the list of animals in a selected province by clicking its area on the distribution map on the encyclopedia page.

Design and Implementation:

- The user clicks a province on the interactive map of China rendered on the encyclopedia page.
- The request is handled by the dispatch servlet and routed to the province controller, which:
 - Retrieves animal records for the selected province from the database;
 - Loads the province's city map data.
- The frontend renders the province page, displaying:
 - A map of the province's cities;
 - A scrollable list of animals with images and descriptions.

As shown in Figure 5, the sequence diagram illustrates this interaction flow in detail.

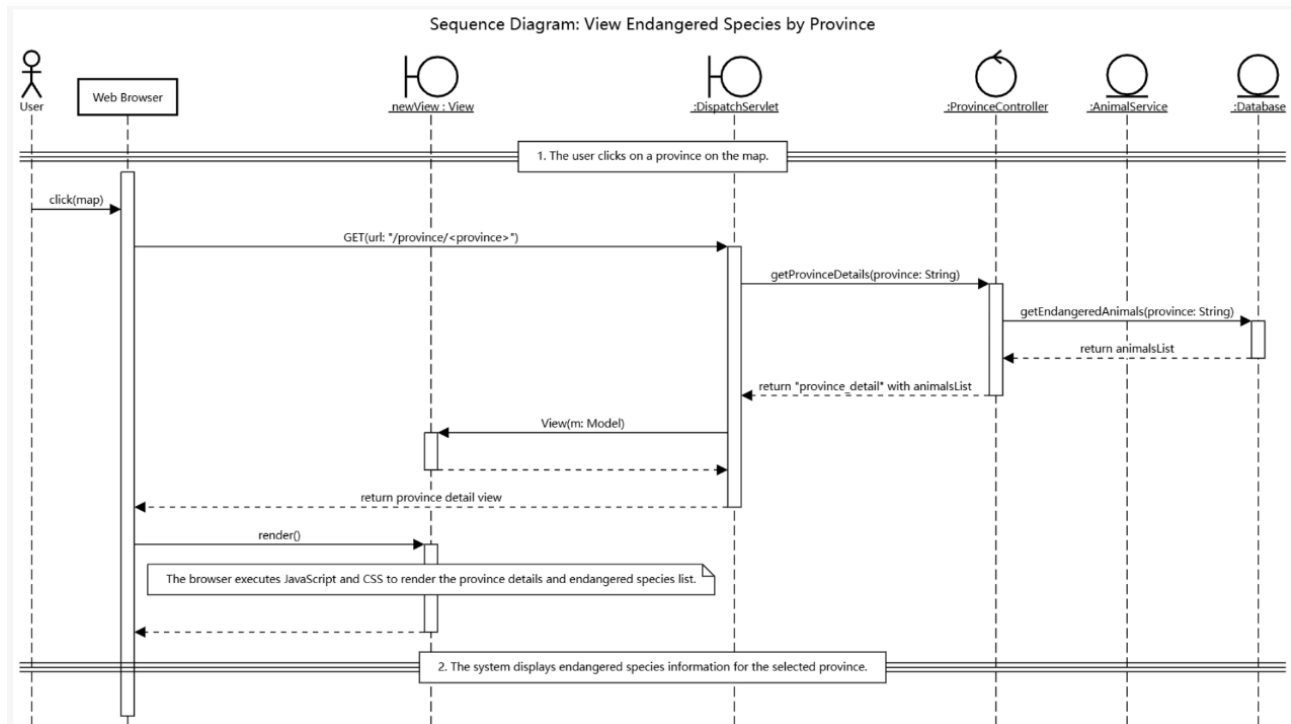


Figure 5: Sequence Diagram of Browsing Animal List via Distribution Map

Event: Playing Matching Game

User Story: User engages with a matching game integrated into the game section of the platform.

Design and Implementation:

- On the game page, the user clicks the “Matching” card to initiate the game session, which navigates to the matching game interface.
- The backend initializes the session by:
 - Randomly selecting 4 species for the current round.
 - Returning each selected species’ name and image URL.
- The frontend receives the data and:
 - Displays a shuffled grid of animal cards, with each species appearing twice.
 - Prepares a countdown timer to define the game duration.

As shown in Figure 6, the sequence diagram illustrates this interaction flow in detail.

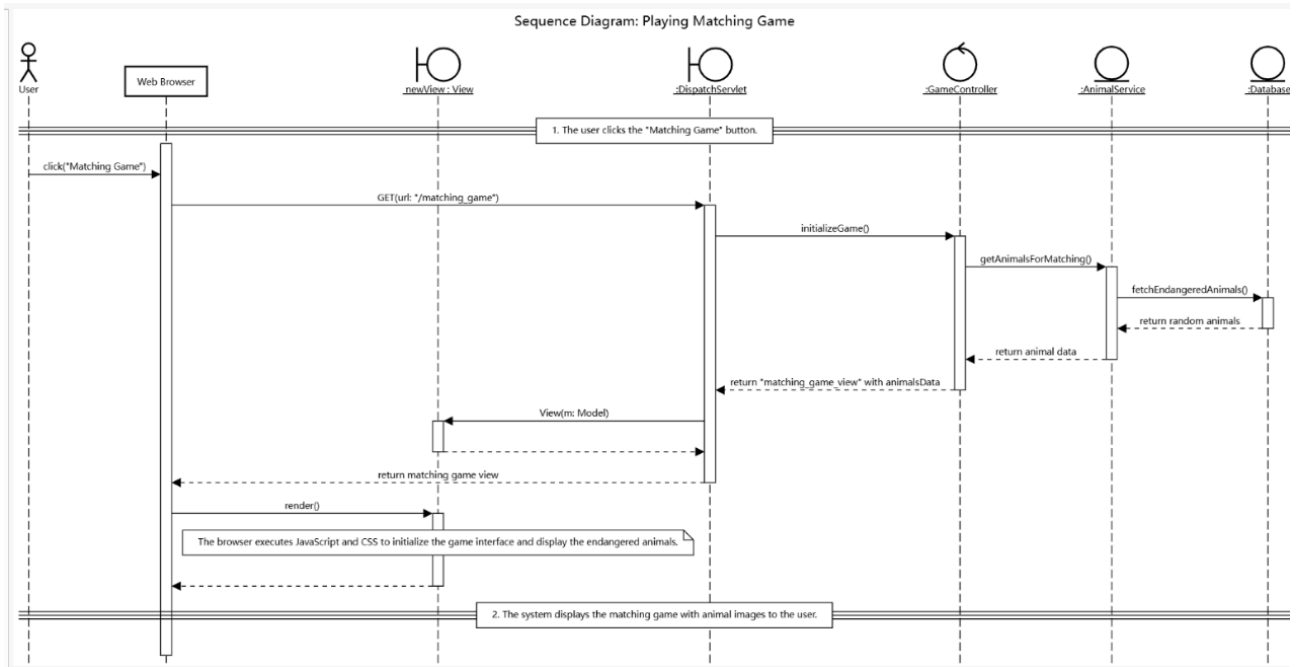


Figure 6: Sequence Diagram of Playing Matching Game

Additional Considerations: User Experience and System Responsiveness

To ensure a seamless and engaging experience, we adopt optimization strategies across three key dimensions: frontend feedback, backend performance, and fault tolerance. These strategies are summarized in Table 3.

Dimension	Strategies
Frontend Feedback	Highlight selected map regions; show loading indicators (spinner, shimmer); animate map transitions.
Backend Performance	Use indexed queries and denormalized lookup tables; apply in-memory caching (e.g., Redis); manage lightweight game session states.
Fault Tolerance	Display user-friendly error messages and retry options; provide fallback content; log detailed error data with context.

Table 3: User Experience Optimization Strategies

2.2.4 System Workflow and Data Processing

The system adopts a unified MVC architecture for handling user requests and rendering responses, as Figure 7:

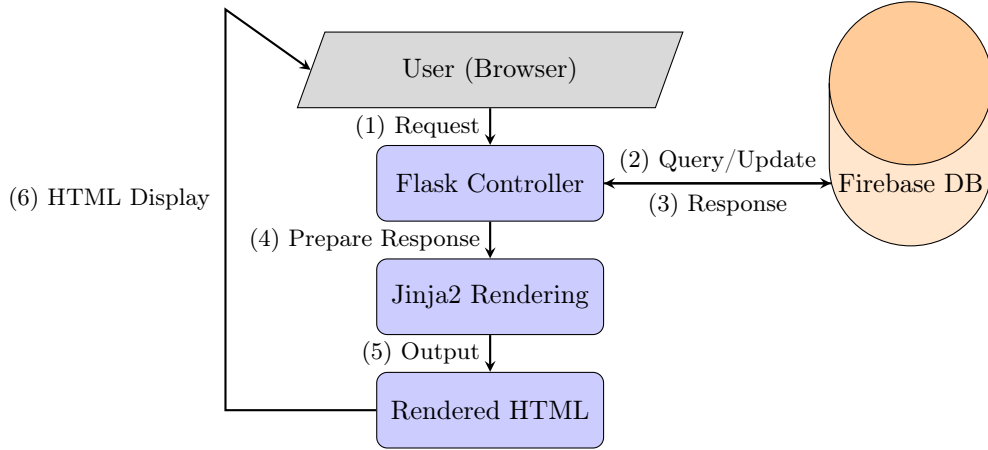


Figure 7: MVC-Based System Workflow and Data Flow

- Users send HTTP requests from their browsers.
- Flask Controller intercepts requests, processes business logic, and queries or updates the database.
- The database returns data to the controller.
- Jinja2 template engine renders HTML pages using the returned data.
- Rendered HTML is sent back to the client browser for display.

Additionally, the system employs APScheduler to run periodic background tasks such as data refresh and maintenance, ensuring the data's timeliness and the system's stability.

- **AJAX-based Asynchronous Interaction:** For interactive components such as real-time quizzes, comment updates, and card drawing, the frontend issues asynchronous requests (via Fetch API) without requiring full-page reloads. This improves user experience and reduces unnecessary rendering overhead.
- **Caching Layer Integration:** To reduce backend pressure and speed up response times for frequently accessed data (e.g., top animals, province game data), a lightweight in-memory caching mechanism is employed. Frequently requested data is stored temporarily in cache (e.g., Flask-Caching or Redis), with periodic refresh controlled by APScheduler.
- **Scheduled Background Jobs:** The system uses APScheduler to periodically refresh key datasets (e.g., fundraising totals, leaderboard rankings) and perform maintenance operations (e.g., stale session cleanup), ensuring data freshness and system stability.

2.3 Data Model Design

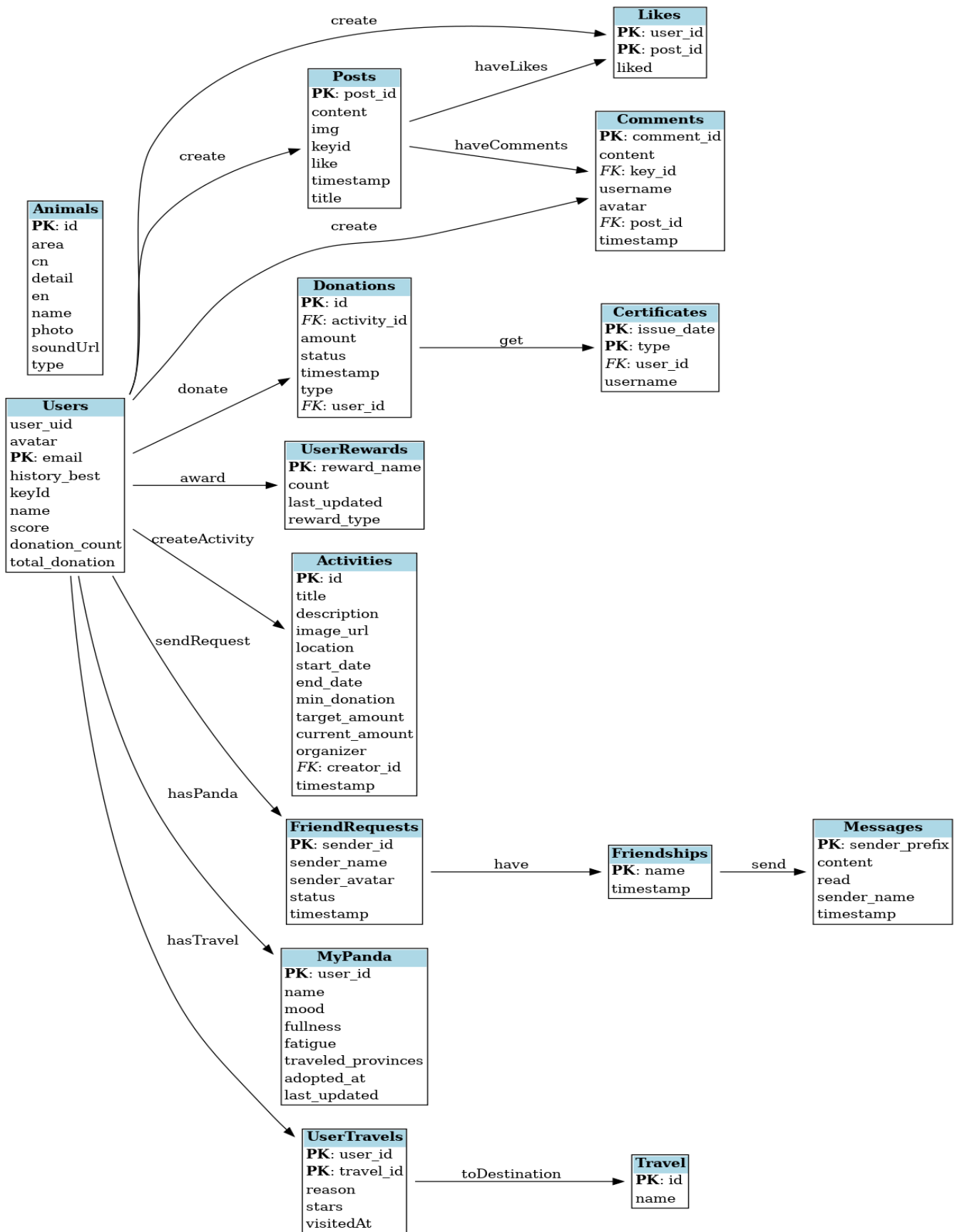
2.3.1 Data Model Overview

The data model of *Dwen Dwen's Neighbour* is designed to support a dynamic, interactive experience with minimal latency and strong consistency. The structure reflects the core entities and their interactions in key features such as encyclopedia, games, my red panda, community and donation.

This modular design supports scalability, denormalized reads, and secure write paths, while enabling real-time synchronization between components.

The following section presents the system's data architecture through an Entity-Relationship (ER) diagram, illustrating how core entities are structured and interconnected to support the platform's major features.

2.3.2 Entity-Relationship Diagram



2.3.3 Data Synchronization and Consistency Strategies

To ensure responsiveness and data integrity, *Dwen Dwen's Neighbour* applies the following Firebase strategies:

- **Real-time Listeners:** UI components reflect updates instantly via Firebase listeners.
- **Atomic Writes:** Multi-path updates ensure transactional consistency.
- **Timestamps for Conflict Resolution:** Server-side timestamps enable last-write-wins logic.
- **Denormalization:** Redundant data (e.g., username/avatar) accelerates reads and is auto-synced.
- **Weak Entity Integrity:** Entities like Likes sync dependently with parent records.
- **Write Isolation:** Scoped writes (e.g., score updates) reduce race conditions.
- **Event-Driven Propagation:** Triggers handle derived updates (e.g., rankings, certificates).
- **Offline Support:** Cached writes sync post-reconnect to preserve user actions.
- **Access Control:** Firebase rules enforce secure, per-path write restrictions.

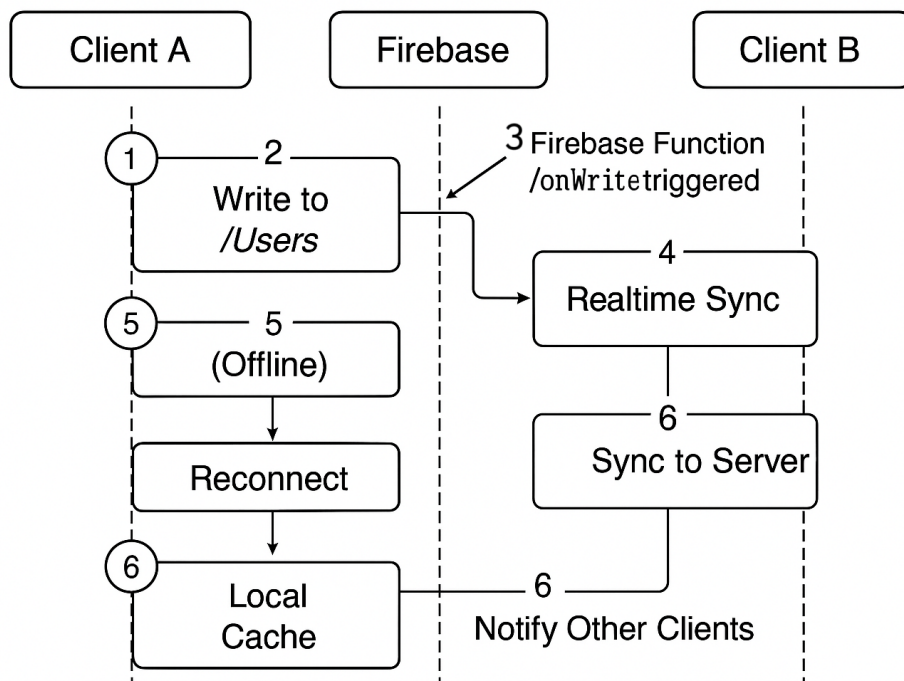


Figure 8: Optimized profile update synchronization flow via Firebase.

Synchronization Flow Explanation (See Figure 8):

1. **Profile Edit:** Client A modifies profile.
2. **Local + Cloud Write:** Change is cached locally and uploaded to Firebase.
3. **Backend Trigger:** `onWrite()` functions process any side effects.
4. **Realtime Sync:** Other clients (e.g., B) receive updates via listeners.
5. **Offline Handling:** Changes made offline sync automatically upon reconnection.
6. **Push Update:** UI on all clients reflects the new data in real time.

These techniques balance real-time consistency with availability, ensuring smooth user interaction even under unreliable network conditions.

2.4 Module Interaction and Communication Design

Dwen Dwen's Neighbour adopts a modular architecture utilizing RESTful APIs and JSON serialization to facilitate inter-module communication, ensuring system cohesion while maintaining component independence.

- **User Identity:** Firebase Authentication is employed to validate user credentials for downstream modules.
- **Gamified Scores:** Real-time score updates are synchronized between the Game and My Red Panda modules.
- **Data Synchronization:** Core data from modules is cached and reused by Profile module for performances.
- **Social Identity:** Community features access centralized user metadata to maintain consistency.

The system uses event-driven notifications to propagate state changes across modules. For example, when a user completes a Challenge game, score updates cascade to both Profile and My Red Panda modules.

API Categories: Key endpoints include Login&Register (/login, /register), Games (/update_score), Community (/get_posts, /toggle_like, /create_comment), and Profile (/update_avatar) and so on.

```
@app.route('/create_comment', methods=['POST'])
def create_comment():
    if 'user' not in session:
        return jsonify({"status": "error", "message": "User not logged in"}), 403

    # Extract data and validate
    post_id = request.form.get("post_id")
    content = request.form.get("content", "").strip()
    # [Additional processing code omitted]

    # Store in Firebase and return response
    comment_ref.child(next_comment_id).set(comment_data)
    return jsonify({"status": "success", "comment": comment_data}), 200
```

Figure 9: Flask Route for Comment Creation

As shown in Figure 9, all endpoints implement consistent error handling with appropriate HTTP status codes. This uniform interface design ensures predictable interactions across the system. In addition, media content is served through Firebase Storage with pre-generated URLs, effectively decoupling storage from application logic and leveraging Firebase's CDN capabilities.

Real-time Data Synchronization: The website leverages Firebase's real-time database to maintain state consistency. When the live status of the red panda changes, updates propagate instantly to all connected clients, as illustrated in Figure 10.

```
// Set up real-time listener for panda attribute changes
function listenForPandaUpdates(userId) {
    // Reference to the user's panda data node
    const pandaRef = firebase.database().ref(`my_panda/${userId}`);

    // Attach listener triggered on data changes
    pandaRef.on('value', (snapshot) => {
        const pandaData = snapshot.val();

        // Update UI with latest data
        updateFeedingControls(pandaData.fullness);
        updateMoodIndicator(pandaData.mood);

        // Trigger health check if critical thresholds are reached
        if (pandaData.fullness < 20 || pandaData.mood < 15) {
            notifyUserOfCriticalState(pandaData);
        }
    });
}
```

Figure 10: Client-side Code for Real-time Data Synchronization

Error Recovery: The system features automatic retries for transient failures with timeouts. Server errors are logged for troubleshooting, while client messages remain clear and actionable, ensuring system resilience.

3 System Testing and Quality Assurance

3.1 Testing Strategy and Implementation

To ensure system reliability, functional integrity, and a high-quality user experience, we implemented a layered testing strategy based on established software engineering practices.

Table 4 summarizes our approach across unit, integration, and functional testing levels.

Test Type	Focus Areas
Unit Testing	<ul style="list-style-type: none">• Authentication System: User registration, login, and session validation.• Animal Data API: Data retrieval, filtering, and presentation.• Firebase Integration: CRUD operations, synchronization, and real-time updates.• Media Processing: Image validation, storage, and upload handling.
Integration Testing	<ul style="list-style-type: none">• Distribution Map: Integration of ECharts with dynamic data.• Web Scraping Pipeline: Data extraction, caching, and Firebase storage.• Community Features: Interaction of posting, commenting, and media upload.• My Red Panda: Adoption flow, attribute updates, feeding, and point logic.
Functional Testing	<ul style="list-style-type: none">• Games: Logic, scoring, and leaderboard functionality.• Donation System: End-to-end donation flow and reward generation.• Friend Chat: Messaging, notifications, and history management.• Extended Features: Language switch, audio playback, and text-to-speech.

Table 4: Summary of Testing Strategy

3.2 Security Architecture

To safeguard sensitive user data and defend against common web vulnerabilities, the system security architecture implements a multi-layered defense strategy to ensure data integrity and system reliability:

- **Authentication and Authorization Management:** Robust user authentication mechanisms are implemented through Firebase Authentication; unauthorized access is intercepted via session validation middleware (@app.before_request); Flask-Session is utilized to ensure secure persistence of user authentication states.
- **Communication Security:** Fine-grained Cross-Origin Resource Sharing control is established through Flask-CORS integration to mitigate CSRF vulnerabilities; standardized JSON formatting and appropriate HTTP status codes are employed to enhance communication reliability.
- **Data Storage Protection:** Sensitive information (e.g., passwords) is encrypted through Firebase security mechanisms; the Firebase Admin SDK enforces strict data access control policies, ensuring backend APIs maintain appropriate permission granularity.
- **Input Validation and Sanitization:** Comprehensive form data validation protocols are implemented to prevent SQL injection and XSS attacks; secure_filename functionality safeguards file upload processes by enforcing strict limitations on permitted file types and sizes to prevent malicious file execution.
- **Exception Handling Mechanisms:** Critical operations are encapsulated within try-except structures for exception capture and processing, ensuring system stability when confronted with unexpected inputs; consistent error response formats are provided across all APIs.

3.3 UX/UI Design Principles

The UI/UX design follows key principles to deliver an intuitive, responsive, and engaging user experience:

- **Visual Consistency:** A cohesive comic-style theme is maintained through unified color schemes (e.g., #FF6B6B as the primary color), standardized card layouts, and consistent use of shadows and border radii.
- **Interactive Feedback:** Micro-animations and instant visual responses (e.g., icon transitions, confetti effects) reinforce user actions and perceived responsiveness; loading indicators communicate system status clearly.
- **Responsive Design:** The interface adapts to diverse screen sizes using flexible layouts and media queries, ensuring optimal usability across desktop and mobile platforms.
- **User-Centered Design:** Navigation is intuitive and streamlined, with clear attention hierarchies, minimized task paths, and contextual guidance such as empty state messages and prompts.
- **Accessibility:** High contrast ratios, touch-friendly controls, and keyboard shortcuts (e.g., ESC to close modals) are implemented to support inclusive interaction.
- **Immersive Experience:** Playful animal motifs, gamified elements (e.g., points, collectibles), and emotionally resonant visuals promote engagement and foster empathy for wildlife conservation.

4 Conclusion

4.1 Future Prospects

Looking ahead, there are several promising directions for future development. One key enhancement could be the introduction of a multiplayer online game mode, enabling users to participate in interactive challenges together. This would not only increase engagement but also foster a sense of community among users. Additionally, organizing offline meet-up events or conservation-related activities in collaboration with local organizations could further strengthen user participation and enhance the platform's real-world impact.

Other potential improvements include expanding the database to include a wider variety of endangered species, adding more dynamic content such as AI assistant and seasonal events, and integrating real-time environmental data to keep users informed of the latest developments in wildlife protection. These enhancements will help sustain long-term user interest and broaden the platform's educational reach.

4.2 Achievements and Reflections

Thanks to the efforts of all team members, *Dwen Dwen's Neighbour* successfully delivered on Week 11, achieving all expected goals with high quality.

The successful delivery of *Dwen Dwen's Neighbour* marks a significant milestone for our company, demonstrating our ability to design, develop, and deploy a functional and impactful system within a limited timeframe. The project met its core objectives, offering a stable, engaging, and educational environment that promotes awareness of endangered species and conservation efforts.

Throughout the project lifecycle, our team worked effectively to overcome technical and logistical challenges, ultimately delivering a well-rounded and scalable solution. Each member's commitment and collaboration were vital to the success of the platform, ensuring that the design met both functional requirements and user expectations.

This experience has deepened our understanding of full-stack development, user-centered design, and agile collaboration. We have learned to prioritize clear communication, adapt quickly to change, and maintain a consistent workflow. These insights will undoubtedly guide us in future academic and professional projects, equipping us with the mindset and tools needed for continuous improvement and innovation.