Wild Snap - Design
Group 29: Sam Eimiller, Aiden Kiefer, Jin Lee, Matt Sagat

Part 3 of the WildSnap report deepens the system-level design and ties it more closely to a realistic implementation and the team's experience building it. WildSnap remains a mobile, gamified app that lets park visitors capture wildlife photos, identify species, view interactive maps, and compete on leaderboards, while giving park rangers tools for restricted zones and moderation. The updated design emphasizes modularity, scalability, extensibility, reliability, and security. Each major feature—capture, mapping, species identification, gamification, and admin tools—is encapsulated in its own subsystem so that one area can change without breaking the rest. Because the app is used outdoors, the design explicitly supports offline caching, queued uploads, and robust error handling. Security and privacy are addressed through role-based access control, encrypted communication, and audit logging, especially for ranger actions and GPS-based data.

The architecture follows a client–server model with a service-oriented flavor. The mobile client handles UI screens (camera, map, leaderboard, profile, admin tools), interacts with GPS and camera hardware, and maintains local caches for offline use. The backend server manages authentication, session state, photo and metadata storage, species recognition integration, restricted zone data, and leaderboard calculations. External services, such as computer vision APIs, mapping SDKs, and cloud databases, are accessed via clear interfaces so they can be swapped or upgraded. Persistent data is organized into logical tables or collections for users, captures, species, leaderboards, achievements, restricted zones, and park boundaries, with images stored in object storage and frequently accessed leaderboard data cached to keep performance acceptable under load. Global control flow is split: the client drives UI behavior (using patterns like MVC or MVVM), while the server coordinates data validation and processing through REST APIs and background sync tasks.

The report organizes the system into six primary subsystems. MapAndLocation renders park maps, wildlife pins, restricted zones, and the user's position, using GPSModule and map data services while supporting offline tiles. CaptureAndIdentification controls the core gameplay loop: PhotoCaptureService manages camera access and offline queues; Capture objects bundle images and metadata; SpeciesIdentifierAPI and SpeciesDatabase handle classification and scoring rules. UserManagement covers registration, login, profiles, permissions, and achievements, distinguishing standard users from AdminUsers. Gamification manages points, ScoreRecord entries, leaderboards, and achievement milestones, feeding updates back into profiles and UI views. SharedServices provides cross-cutting infrastructure like photo storage, cloud

synchronization, and LoggingService. AdminTools lets authorized rangers create and validate restricted zones, moderate or correct content, and feed updates into MapAndLocation, all under strict authentication and auditing.

The report is explicit about new problems, environmental constraints, and risks introduced by WildSnap. Continuous GPS tracking and detailed photo locations raise privacy concerns, and gamified scoring may unintentionally encourage risky behavior as users chase rare species or high-value locations. Limited connectivity, bad weather, battery life, and GPS drift all affect usability and accuracy in real parks. Staff may see extra workload moderating photos, managing restricted zones, and responding to in-app reports, and existing network or back-office systems may need modest upgrades to handle increased traffic. Technically, the system faces risks such as high inference costs for species identification, map API rate limits and licensing, growing cloud storage bills, security threats to admin tools, and potential GPS spoofing to cheat the game. Longer-term, the team must guard against feature creep, the burden of supporting many device and OS versions, and the ongoing need for maintenance and model retraining.

Cost and planning considerations are also summarized. The estimated initial budget is about $50,000 for design, development, infrastructure, API usage, testing, and basic marketing, plus the non-financial cost of hundreds of hours over roughly a year of work. Ongoing costs include hosting, storage, API fees, and updates, along with opportunity cost: time spent on WildSnap cannot go to other projects or research. “Waiting room” ideas for future versions include augmented reality overlays, deeper conservation database integration, group or seasonal challenges, offline on-device models for remote parks, and richer educational content and analytics to fine-tune the experience. The project retrospective closes this part by reflecting on process: dividing work by responsibility helped keep tasks manageable, but working in “bubbles” sometimes caused last-minute mismatches and rushed fixes. For future projects, the team would keep clear responsibility splits but add frequent check-ins, shared naming and formatting conventions, and more explicit planning. In that sense, CS 440 taught the group as much about communication and coordination as it did about system design and architecture.