
Project Plan Document: Pet Authentication and Welfare System (PAWS)

Team ID: PES1UG22CS265_PES1UG22CS277

Team Members:

1. K R Rakshith (SRN: PES1UG22CS265)
2. Karthik H Kademani (SRN: PES1UG22CS277)

Date: 02/10/2024

---

1. Project Lifecycle Model

Incremental Model

Description: The Incremental model develops the project in small, functional increments, where each increment adds new features. Each iteration builds upon the previous one, allowing you to focus on core functionalities first.

- **Pros:**
    - **Flexibility in developing key features first, then adding enhancements incrementally.**
    - **Allows early deployment of a basic version.**
    - **Easier to manage scope within a short time frame.**
- **Cons:**
    - **Managing multiple versions and iterations can be a challenge for a small team.**
    - **Requires clear prioritization of core features.**

Why it works for us: With a tight timeline, we can prioritize critical components (e.g., Pet Registration and Medical Records) and gradually add features (e.g., Insurance) as increments. This approach ensures that even if time runs out, we'll have a functional core system.

---

2. Tools for Lifecycle Execution

Planning Tools:

- **Jira :** Task management and sprint planning.
- **GanttProject/Excel:** Creating and managing the Gantt chart for the project timeline.

Design Tools:

- **BoardMix:** For creating ER diagrams, data flow diagrams (DFD), and workflow designs.
- **Figma/Adobe XD:** For wireframing the user interface (UI).

Version Control:

- **GitHub/Git:** Version control for source code, ensuring smooth collaboration and tracking changes.

Development Tools:

- **Backend (Business Logic Layer):** Direct interaction between Streamlit and the database using Python libraries like SQLAlchemy or psycopg2 to handle database operations.
- **Frontend (Presentation Layer):** Developed using Streamlit, a Python-based web framework that allows for quick prototyping and building of functional web apps
- **Database: MySQL/PostgreSQL**: Relational database for storing and managing pet, owner, and medical data.

Bug Tracking:

- **JIRA/GitHub Issues:** Tracking bugs, issues, and features throughout development.

Testing Tools:

- **Postman:** For testing the API endpoints.
- **Selenium:** For automated testing of the user interface.
- **JUnit/Mocha/Chai:** For unit testing of the backend and frontend components.

---

3. Deliverables

Reuse/Build Components:

Reuse Components:

- **Authentication Module:** Reuse of available authentication libraries (OAuth, JWT) to implement secure login functionality.
- **Database Management System:** Pre-existing relational databases like MySQL/PostgreSQL.

Build Components:

- **Pet Registration System:** Custom module for registering pets with unique identifiers.
- **Medical Record Management:** Custom medical record and vaccination tracking system.

Justification:

- **Reuse of authentication libraries** reduces development time for login security, as they provide reliable, pre-tested solutions.
- **Custom components** for pet registration, medical records, and adoption are necessary since PAWS is a unique system tailored to managing pet welfare, which requires specific logic and workflows.