**GROUP NUMBER: 15**

**1. PROJECT TITLE**

**Smart Pet You Care About Tons (SPYCAT)**

**2. GENERAL PROJECT DESCRIPTION**

**SPYCAT is a virtual pet system built with a De1-SoC FPGA board. Its aim is to improve the user’s quality of life by providing companionship and virtual interaction with friends.**

**The virtual pet will live inside the VGA monitor and be animated and output sound according to the pet's states and respond to user interactions. There are clear visual differences between SPYCAT in sleeping state, eating state, and regular state. There will be no audio output during the sleeping state of the pet and music will start playing when the pet is awakened.**

**Different UI components on a touchscreen connected to the De1 board will provide user interactions with the pet such as feeding, petting and playing. SPYCAT will also detect when the user is around via motion detection to increase user immersion - motion detected will wake the pet from sleep and it will return to sleep after a configured amount of inactivity.**

**There is an authentication system which users can use to register their pets and login to their account. During registration, users can configure their pet’s colour and their location. Different pet colours are used to distinguish different users’ pets during multiplayer interactions. Location information is used to fetch accurate weather data to further increase user immersion. The pet’s mood and the ambient background will match the current weather. (Eg, pet is sad when it is raining and background changes to rainy)**

**There is multiplayer functionality where the user can connect with their friends’ SPYCATs. The friend system is similar to social media platforms where users can add/ delete friends, send messages, and take their pet on playdates with their friends. A playdate consists of the other user’s pet showing up on this user’s product, where this user can then pet, feed, and play with both pets. Playdates also include a mini-game, namely rock-paper-scissors for the users to play against each other using their pets as sprites. The winner of the game will gain friendship points which are stored in a cloud database, then used to build a leaderboard on a web app showing which users have the highest friendship points in real time.**

**3. TARGET MARKET**

**targeted towards teens aged 14-18**

**4. NEEDS AND CONSTRAINTS OF TARGET MARKET**

1. **Ease of use: Less physical buttons use and use touchscreen**
2. **Portable: The system is built in a compact size**
3. **Audio output: announce pet status and message notification alerts**
4. **Adjustable audio: audio volume can be adjusted by touchscreen buttons**
5. **High resolution screen with large text and graphics**
6. **Internet connection - use wifi hardware module**
7. **Low cost**

**5. COURSE REQUIREMENTS**

**Hardware**

* **Audio output (Speaker) - 3.5mm audio port**
* **Camera input (motion detection)**
  + **GPIO camera**
  + **on/sleep functionality**
* **Touch Screen - AR1100 controller attached to UART**
* **LCD - VGA output port**

**UI**

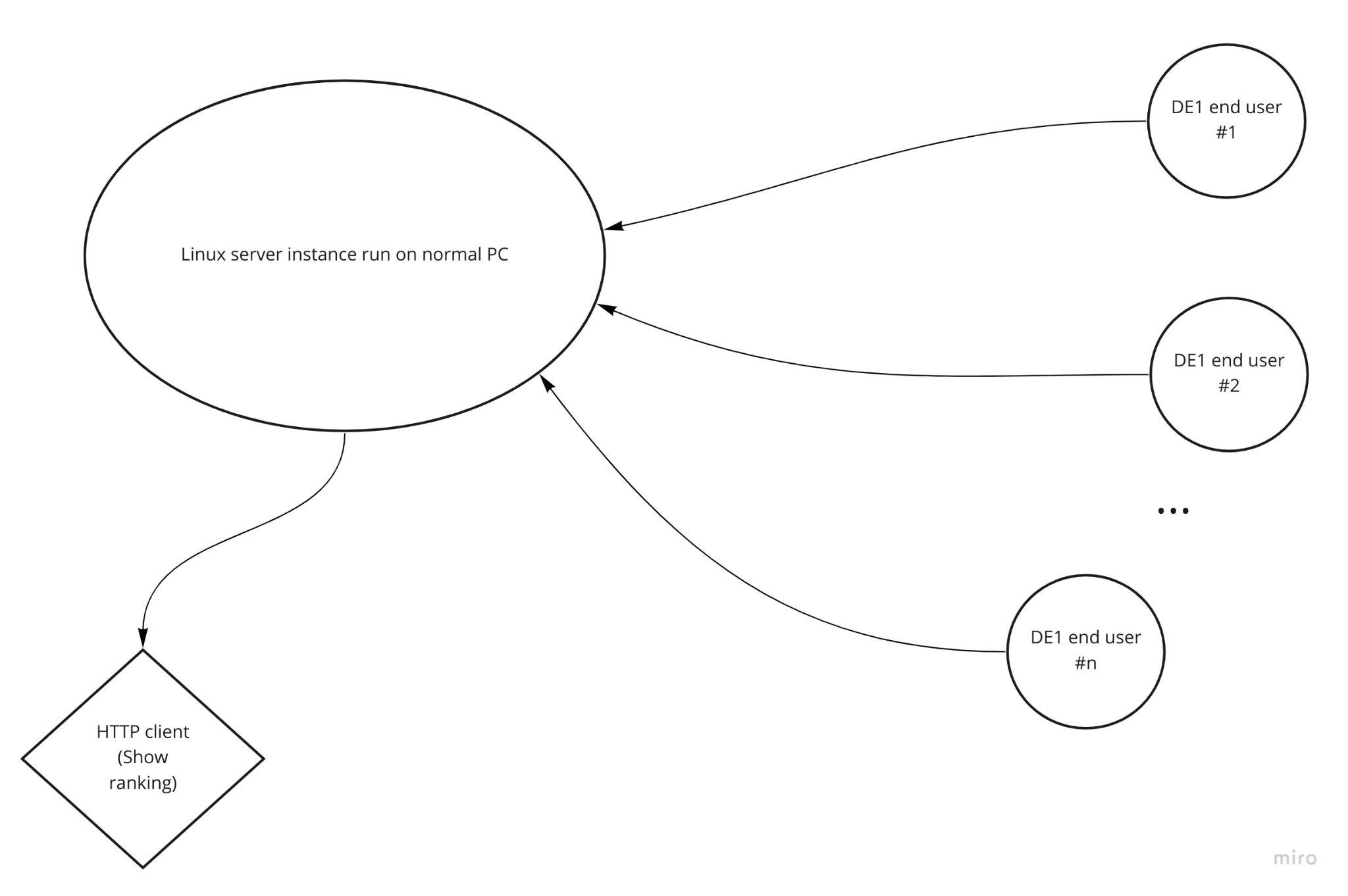
* **Hardware:** 
  + **LCD - VGA output port**
  + **Touch screen - AR1100 controller attached to UART**
  + **Speaker - 3.5mm audio port**
  + **Camera - GPIO**
* **Function:**
  + **Show feedback and pet on screen**
  + **Show user ranking data on the web**
  + **Authentication system by login and register using user id from database**
  + **Friends list management panel: Add/delete by searching user ID**
  + **Real time chat function between friends**

**Cloud**

* **Cloud runs on Linux server**
* **Friend system - add friends online by user ID**
* **Provide real time chat services between friends**
  + **They communicate by an on-screen virtual keyboard and chat UI**
* **Fetch real time weather data to display appropriate ambient background and pet mood**
* **Front-end website run on Django framework that shows a real time leaderboard of SPYCAT users with the most friendship points**
* **User contents stored in MongoDB back-end database**
  + **Store user data (user id, pet colour, and accumulated friendship points)**
  + **Store friends list of each players**
  + **Store the chat content**

**Network**

* **Hardware**
  + **Wifi module**
* **DE1-SoC Wifi connection is required to connect to server**
* **DE1-SoC runs Linux instance which has a provided tcp/ip network stack**
* **Server works as a back-end and front-end server to retrieve the user performance score and save to the back-end and showing the multi-user ranking data on the front-end website.**

****

**Societal impact**

* **Especially during the pandemic, quarantine can easily lead to people feeling lonely. Having a smart pet would not only improve quality of life by making mundane tasks easier for the user, but also help prevent mental health levels from dropping due to loneliness.**

**6. FEASIBILITY OF HIGH-LEVEL DESIGN AND RISK MITIGATION STRATEGY**

**The De1-SoC will be hosting the main local components of this Smart Pet device. It will be running a Linux Instance on the ARM-based hard processor system which allows us to use Wifi to connect to the Linux server to send/ grab data. It will also be configured to allow parallel custom verilog drivers to interface with our peripherals (touchscreen, VGA Display Out, Speaker, and Camera).**

**The combination of De1 and the linux server run on PC will allow us to provide features that rely on cloud. The linux server will receive user input information from the De1 and call appropriate functions to store them in our Mongo database via an API call. The server will also fetch data from the database and from OpenWeather API and push the information to the De1 via tcp/ ip connection on Wifi.**

**To capture diversity and inclusion, we include multiple ways the user can interact with the virtual pet. As an example, we include both motion detection and touch screen interaction as ways to wake the pet. We will also ensure we choose a colour scheme that is colour blind friendly.**

**For the motion detection component, we will be using a GPIO camera connected to a linux instance on the De1. The image input will be passed to python code running OpenCV to detect motion. This step is algorithmically complex, so as a backup we will use image similarity algorithms to imitate motion detection. In this case, our product does not depend entirely on the functionality of pre-built motion detection models. Our product can still work by taking snapshots at set time intervals and an image similarity algorithm such as ​​keypoint matching to imitate detecting motion.**