# **25-Week Implementation Plan: Pepper Medical Assistance Robot Project**

## **Weeks 1–2: Project Kickoff and Initial Setup**

* **Week 1 (Initiation):** Kick off the project with a team meeting to finalize requirements, the system architecture, and role assignments. **Aly (Team Lead)** sets up the Pepper robot hardware (updates firmware/NAOqi, ensures Pepper connects to the lab/hospital Wi-Fi) and verifies basic functions (movement, camera, microphones). **Abdulrahman** and **Mohammed** begin designing the Pepper tablet’s Android-based UI – sketching wireframes for the bilingual interface (tabs for Check-in, FAQ, Triage) and planning how to deploy this UI on Pepper’s tablet via the ALTabletService API. **Amira** starts researching Natural Language Understanding tools for Arabic/English and compiles a list of frequently asked patient questions (FAQ) in both languages to build the knowledge base. **Amr** reviews clinical triage guidelines (e.g. ESI criteria) and gathers requirements for a triage severity model (what symptoms to ask, scoring rules). **Youssef** assists Amira in collecting sample Q&A pairs from Andalusia Hospital resources and sets up a shared repository for data and code, ensuring all team members have the development environment configured (Pepper SDK, FastAPI, Whisper, etc.).
* **Week 2 (Environment Setup):** Establish the development environments for each subsystem. **Aly** configures Pepper’s software environment – installing the NAOqi SDK (Python 2.7) on a laptop and on Pepper, and testing a simple “Hello, world” program (Pepper speaks a phrase using ALTextToSpeech via a short script). He also creates a basic map of the area or testing space for Pepper to enable future navigation. **Abdulrahman** scaffolds the backend server using FastAPI (Python 3), creating placeholder REST endpoints such as /api/speak (to send text for Pepper to speak) and /api/faq/search. He ensures the Pepper tablet can load a simple webpage through Pepper’s tablet service. **Mohammed** builds a basic static web page for Pepper’s tablet (the **pepper\_ui**): an index.html with navigation tabs for **Check-In**, **FAQ**, and **Triage**. This page is designed with large text and buttons for hospital users. He verifies Pepper’s tablet can display this page (e.g. by hosting it on the backend and using Pepper’s ALTabletService to open the URL). **Amira** sets up Whisper (speech-to-text) on a development PC and runs quick tests with Pepper’s microphone: Pepper records a short voice clip and Amira uses Whisper to transcribe it[mdpi.com](https://www.mdpi.com/2313-7673/9/7/391#:~:text=Whisper%20,speech%20recognition%20model%20that%20has), confirming that Arabic and English speech are recognized. **Amr** drafts a triage questionnaire outline (e.g. yes/no questions about critical symptoms) and a simple rule-based scoring method (e.g. assign points for symptoms) to be implemented in code. **Youssef** begins organizing the FAQ data (compiling the questions Amira gathered into a small FAQ database or CSV) and installs a vector database or text search library in the backend for the Retrieval-Augmented Generation (RAG) module. By the end of Week 2, the team has a clear plan and a running development setup – Pepper is connected, the backend server runs with dummy endpoints, and the tablet can display a test page.

## **Weeks 3–5: Core System Development Begins**

* **Week 3 (Backend & UI Foundation):** Start implementing core functionalities in parallel. **Abdulrahman** fleshes out the FastAPI backend: defines data models and stubs for key API routes (/api/faq/search, /api/triage/submit, etc.), and sets up WebSocket connectivity (using bridge\_bus.py) to Pepper for sending commands. He ensures that a test message from the server can make Pepper speak (e.g. calling the /api/speak endpoint publishes a TTS command that the Pepper **bridge.py** on the robot will execute, making Pepper say a test phrase). **Mohammed** and **Abdulrahman** collaborate on the Pepper tablet UI: Mohammed implements the front-end HTML/CSS for the check-in form (patient can enter or select their ID/appointment), an FAQ query text box, and a triage questionnaire page with yes/no buttons. Abdulrahman links the front-end to backend by writing JavaScript (script.js) to fetch() the backend APIs when the user submits a question or form. **Aly** works on Pepper’s robot-side code (**robot/bridge.py**): he writes a WebSocket client that connects Pepper to the backend server. This bridge will listen for messages like {"type": "tts", "text": "Hello"} or {"type": "tablet", "url": "http://.../index.html"} and call the appropriate NAOqi functions (ALTextToSpeech, ALTabletService) on Pepper. Aly tests the connectivity by sending Pepper commands from the backend (e.g. make Pepper wave or speak on command). **Amira** begins implementing the **FAQ retrieval (RAG)** module: she uses the collected Q&A data to build an embedding index (writing a script to encode FAQ questions and store vectors) and codes rag.py to query this index for the best answer to a user’s question. For now, this may be a simple keyword or embedding search returning a canned answer from the database. **Amr** starts coding the **triage logic** in triage.py: based on his Week 2 outline, he creates functions to calculate a triage score from yes/no symptom inputs (e.g. assign severity points for each “yes” answer). He keeps it rule-based initially (e.g. if chest pain = yes and difficulty breathing = yes, score = high severity). **Youssef** supports both Amira and Amr by gathering any additional data they need (e.g. collecting more FAQs from hospital pamphlets or ensuring triage rules align with medical standards) and sets up test cases – e.g. a few example questions to test the FAQ module and sample symptom profiles to test the triage scoring function.
* **Week 4 (Integration of Voice & UI Prototypes):** **Aly** and **Amira** focus on the voice interaction pipeline. Aly writes a Pepper **listen** function: Pepper’s microphones capture audio (via NAOqi ALAudioRecorder), and the audio stream or file is sent to the backend for Whisper processing[mdpi.com](https://www.mdpi.com/2313-7673/9/7/391#:~:text=Whisper%20,speech%20recognition%20model%20that%20has). Amira integrates this into the backend: possibly by adding an /api/speech endpoint or extending /api/faq/search to accept audio. They test the end-to-end pipeline: Aly triggers Pepper to listen when a patient presses a “Voice Input” button on the tablet (or via voice command), the audio is transmitted to Amira’s Whisper module, transcribed to text, and then passed into the FAQ search. They iterate until Pepper can reliably capture short queries (within ~1 meter distance per best practice[mdpi.com](https://www.mdpi.com/2313-7673/9/7/391#:~:text=by%20finding%20the%20maximum%20accuracy,the%20comprehension%20of%20spoken%20statements)[mdpi.com](https://www.mdpi.com/2313-7673/9/7/391#:~:text=discarding%20data%20points%20with%20values,to%20one%20meter%20from%20the)) and display or speak the answer. **Abdulrahman** improves the UI based on initial tests: ensures that when Pepper speaks an answer, the text also appears on the tablet for clarity (subtitles) – for this, he utilizes the WebSocket bridge to receive the answer text and updates the web UI via JavaScript (e.g., showing the answer on screen). **Mohammed** continues developing the UI/UX: refining the check-in page (e.g., a form to enter patient name or select from a list of doctors) and making the interface bilingual (adding Arabic text alongside English, with a language toggle if needed). He also starts looking at the **Andalusia hospital mobile app** or notification system for nurse alerts – reaching out to hospital IT if needed to understand how an alert could be integrated (e.g., via an existing nurse app or an SMS gateway). **Amr** refines the triage model, possibly beginning to code a simple **ML classifier** if data permits (for example, a decision tree or logistic regression using synthetic patient data). If no dataset is available, he continues with the rule-based approach but structures the code for easy adjustments. **Youssef** sets up a database or persistence for any required data (e.g., a SQLite or simple JSON for storing check-in info and triage results) and helps Mohammed prepare a dummy *nurse alert* mechanism (for instance, a placeholder that logs an alert or sends a notification email when an urgent case is detected). By end of Week 4, the team has a basic voice-query capability working in the lab and the UI is taking shape.
* **Week 5 (Testing Core Components):** This week is about verifying that all the fundamental pieces work in isolation before merging them. **Aly** tests Pepper’s autonomous navigation basics: he uses Pepper’s navigation API to have Pepper roam a small area or approach a waypoint, ensuring that the robot can move safely if needed for the demo (e.g., Pepper can turn to face the user or move a few meters autonomously). He also ensures Pepper’s tablet displays the UI and Pepper’s voice output is audible in a noisy environment (simulating a hospital lobby). **Abdulrahman** writes unit tests or simple scripts for the backend APIs (FAQ search returns an answer for test queries, triage submit returns a score as expected, etc.) and fixes any bugs in the request/response handling. **Mohammed** conducts a UI usability review – checking that buttons are large and labeled clearly in Arabic and English, and that the workflow is intuitive (e.g., after check-in, it goes to FAQ or directly to triage if needed). He might recruit a classmate to pretend to be a patient and get feedback on the interface. **Amira** tests the NLU pipeline with a small set of sample questions in both languages (e.g., “What are the visiting hours?” in English and Arabic) to tune the FAQ retrieval – for instance, she might expand the FAQ database or tweak the embedding search if Whisper’s transcription has certain quirks with Arabic dialect. **Amr** validates the triage logic with a medically knowledgeable person (if available) or uses common sense scenarios to ensure the scoring makes sense (e.g., a patient saying yes to all critical symptoms yields a high-risk score). **Youssef** consolidates all test findings into a report and schedules an integration session for next week – the team will start integrating components, so he ensures everyone’s code is pushed to a common repository and that environment setup instructions are updated for consistency. By the end of Week 5, the project has solid individual modules (Pepper control, UI, voice recognition, FAQ retrieval, triage logic) that are ready to be integrated.

## **Weeks 6–10: Integration & Mid-Project Demo Preparation**

* **Week 6 (First Integration Pass – Pepper & Backend):** The team begins integrating the robot with the backend services. **Aly** deploys the **bridge.py** client on Pepper and runs the FastAPI server on a local machine; they establish persistent communication. The goal this week is that Pepper, the backend, and the tablet UI all talk to each other. **Abdulrahman** helps integrate the Pepper tablet UI with the backend via the WebSocket bridge: e.g., when a user taps a button on the tablet UI (which calls a backend API), the backend uses bridge\_bus.py to send a command to Pepper and Pepper executes it. One test scenario: the patient enters an ID on the tablet, the backend receives it and publishes a “speak” command, and Pepper says “Welcome [Name], your appointment is confirmed” (a simulated check-in). This tests end-to-end flow for the check-in feature. **Mohammed** finalizes the check-in workflow for the demo – possibly using a fixed list of example patient names/appointments for simplicity – and ensures the UI transitions smoothly from check-in to other functions. **Amira** integrates the speech recognition fully: she connects the Whisper transcription component to the live system. Now, when Pepper’s mic is activated (e.g., user says “Pepper” or presses a “Speak” button), the audio is sent to the backend, transcribed by Whisper, and the text fed into rag.py for answer retrieval. She works on automating this pipeline so it’s user-friendly (for the demo, they might allow a push-to-talk interaction on the tablet). **Amr** connects his triage scoring to the system via the /api/triage/submit endpoint – for now, the triage might not be demonstrated live, but he ensures that when given sample yes/no answers, the backend can compute a triage level. **Youssef** coordinates this integration testing: he plays the role of a patient to run through scenarios (check-in, ask a question) while the others observe and debug. The team identifies and fixes integration issues (e.g., CORS issues in web requests, timing delays between Pepper and server, etc.). By end of Week 6, Pepper should be **minimally interactive**: able to take input (via tablet or voice) and produce an output (speech and on-screen text) through the integrated pipeline.
* **Week 7 (FAQ QA & Bilingual Support):** Focus on polishing the FAQ feature in preparation for the demo. **Amira** expands the FAQ knowledge base using a **Retrieval-Augmented Generation (RAG)** approach: she ensures rag.py can handle a variety of phrasings by perhaps adding more Q&A pairs or using a similarity search so that even if the question is not an exact match, Pepper can find a relevant answer. She also works on bilingual capability: verifying Whisper can detect Arabic questions and that the system can retrieve Arabic answers if the query is in Arabic. If needed, she implements language detection or simply checks Whisper’s language output and selects the appropriate answer language from the FAQ database. **Aly** updates Pepper’s text-to-speech settings to support both languages – for example, using Pepper’s API to switch the voice/language to Arabic when neededdoc.aldebaran.com. He tests Pepper speaking a few Arabic answers (like greeting or common question responses) to ensure clarity and correct pronunciation. **Abdulrahman** and **Mohammed** refine the UI to handle longer answers and multi-language content: for instance, ensuring that Arabic text (RTL script) is properly displayed on Pepper’s tablet and that the font is readable. They also add a visual indicator on the UI when Pepper is listening or processing a voice query (e.g., a microphone icon or spinner) to improve user experience. **Amr** and **Youssef** work in parallel on a small evaluation of the FAQ module – Youssef collects 5–10 sample questions (some in Arabic, some in English) and their expected answers from hospital materials, then they test Pepper’s responses. They log any failures or misrecognitions to guide further tuning (for example, if Whisper mishears a particular medical term, they might consider adding a custom vocabulary or a correction filter in code). By the end of Week 7, the FAQ feature should be robust in both Arabic and English, ready to demonstrate Pepper answering common patient questions by voice and tablet.
* **Week 8 (Triage Prototype & Nurse Alert Planning):** With the FAQ and basic check-in working, the team turns attention to the triage and alert features. **Amr** develops a prototype triage questionnaire flow on the Pepper tablet: he creates a sequence of yes/no questions in the UI (or possibly Pepper verbalizes them one by one). Initially, this can be a simple checklist on the tablet (e.g., “Do you have chest pain? [Yes/No]”) that the patient can tap. He connects this to the backend /api/triage/submit so that once all questions are answered, the backend computes a triage score or category. **Mohammed** assists by making the triage UI user-friendly (e.g., one question per screen or a progress bar if multiple questions). Meanwhile, **Youssef** and **Mohammed** flesh out the **nurse alert** mechanism: they design how an alert will be triggered when a high-risk triage result is detected. This could involve the Pepper robot itself (e.g., Pepper saying “Please assist, urgent case!” and flashing lights) and an external alert to hospital staff. Mohammed checks if the Andalusia Hospital’s existing mobile app or system can receive external triggers – if so, they plan to integrate via an API or webhook. If not, they decide on an alternative (perhaps sending an SMS or a notification through a custom lightweight app). **Aly** and **Abdulrahman** integrate a preliminary alert on Pepper’s side: using the bridge to make Pepper perform an “alert” action when instructed. For example, they program Pepper to raise an arm LED or play a sound along with a vocal alert. **Amira** focuses on ensuring the chatbot can handle the triage context if needed – possibly not needed if triage is mostly button input, but she ensures that if a patient asks a question during triage, the system can either handle it or politely defer. The team by end of Week 8 has a basic triage questionnaire implemented and a clear plan for nurse alerting (with Pepper’s on-robot alert working and a strategy for notifying staff externally).
* **Week 9 (Integration Testing – Demo Rehearsal):** Full end-to-end rehearsal of the mid-project demo features. The team treats this week as a *soft freeze* on new features and focuses on stability. They integrate all pieces into a cohesive demo script. **Aly** oversees a dry-run where a team member acts as a patient going through the entire interaction: arriving at Pepper, checking in, asking a question, and (optionally) going through triage. All team members run the system together to observe any issues. **Abdulrahman** monitors the backend during the test, checking for any errors or slow responses, and fine-tunes performance (e.g. if Whisper is too slow, they might switch to a faster model or pre-record the query for demo purposes). **Mohammed** ensures the UI responds correctly at each step (e.g., after check-in, Pepper’s tablet maybe automatically switches to FAQ tab – he can implement such logic via the ALTabletService command to change the view). **Amira** refines the dialogue flow: for instance, making Pepper’s spoken responses more natural (she might update some answer phrasings or add a polite preamble like “According to the hospital policy, visiting hours are 9am to 9pm.”). **Amr** verifies that if they decide to showcase triage in the demo, the logic triggers the alert properly. He might simulate an *urgent patient* scenario to ensure Pepper indeed gives the nurse alert line and that the backend prepares whatever notification (though the actual mobile push might not be shown live, Pepper’s voice alert will be). **Youssef** acts as the coordinator: he times the run-through to ensure it fits presentation limits, notes any part where the system lags or might need explanation, and compiles a checklist of fixes. By the end of Week 9, the team resolves last-minute bugs (for example, ensuring Pepper doesn’t accidentally time-out or that the volume is adequate) and finalizes what each member will present or demonstrate.
* **Week 10 (Mid-Project Demo and Presentation):** **Milestone: Mid-Project Presentation.** The team delivers a working demo to faculty/peers, showcasing the Pepper robot with partial features implemented. In the demo scenario, a patient approaches Pepper: Pepper greets them and goes through a **basic check-in** (the user enters their name or ID on the tablet, and Pepper responds verbally with a confirmation). Next, the patient asks Pepper an **FAQ** (“Where is the pharmacy?” or “What are visiting hours?”) using voice; Pepper (with Whisper’s help) recognizes the question, retrieves an answer from its knowledge base, and replies **in the corresponding language**, both **speaking it and displaying the text** on the tablet. These features (speech recognition, bilingual response, on-screen info, and Pepper’s connectivity) form the core of the mid-project demo. Each team member takes on a role during the presentation:  
  + **Aly** introduces the project scope and live-demoes Pepper’s capabilities, narrating what Pepper is doing (he also operates Pepper if any manual control is needed). He highlights Pepper’s hardware integration and connectivity (e.g., “Pepper is connected to our server and responds in real-time”).
  + **Abdulrahman** explains the system architecture and backend integration. He describes how the Android tablet UI on Pepper communicates with the FastAPI backend and how commands from the server make Pepper talk or display information. He may show the tablet interface to the audience and the network calls happening behind the scenes.
  + **Mohammed** focuses on the user interface and user experience. He walks through the tablet UI’s design (bilingual text, large buttons) and demonstrates the check-in form and how a user would navigate the FAQ and triage tabs. He might also mention plans for integrating the solution with the **Andalusia mobile app** for nurse alerting.
  + **Amira** discusses the **NLU and voice interaction**. She explains how Pepper understands speech using Whisper and a chatbot approach, supporting both Arabic and English queries. She can present an example of the FAQ retrieval working (perhaps by asking Pepper an unscripted question to show the robustness).
  + **Amr** covers the **triage severity detection** component. He outlines how the triage questionnaire works and the rule-based (or ML) model that scores patient responses. Even if the triage feature is only partially implemented by week 10, he presents the design and maybe a short recorded clip or simulation of Pepper asking a health question and reacting to an “urgent” case.
  + **Youssef** wraps up with project management aspects and next steps. He highlights what has been achieved by mid-project (Pepper can greet, answer FAQs by voice, etc.) and what is coming next (full triage and nurse alert integration, more testing, deployment in hospital). He also mentions any preliminary user feedback gathered and the plan for final testing at Andalusia Hospital.
* The mid-project presentation is a success – the prototype Pepper assistant impresses the audience by checking in a patient and answering questions. The team notes any feedback (e.g., improve speech speed, or add more FAQs) to incorporate going forward. **Milestone achieved: a working demo by Week 10.**

## **Weeks 11–16: Feature Expansion and System Integration**

* **Week 11 (Post-Demo Debrief & Planning):** The team meets to discuss feedback from the mid-project demo and to plan the second phase of development. They prioritize tasks for remaining features: robust triage interaction, nurse alert integration, Pepper’s navigation, and extensive testing. **Aly** updates the project plan and assigns tasks ensuring everyone’s responsibilities are clear for the next milestones. Based on feedback, **Amira** may plan to improve the NLU (for example, if the demo revealed any misheard queries or language issues, she’ll address those by adjusting Whisper settings or adding more FAQ data). **Amr** uses this week to incorporate any advice on the triage model (perhaps simplifying or expanding the questionnaire based on mentor input). **Mohammed** begins deeper integration with the hospital systems: he reaches out to Andalusia’s IT department (if not done already) about how Pepper’s system could send an alert to nurses – for instance, via their existing **Andalusia mobile app** or any internal messaging system. **Youssef** starts designing a formal **usability testing plan** for the final deployment (which patient scenarios to test, what feedback to collect, success metrics like response time or user satisfaction). The outcome of Week 11 is a refined roadmap for weeks 12–16 and a clear set of improvements to implement, ensuring the team addresses any weaknesses identified at mid-term.
* **Week 12 (Triage Conversation & Pepper Behavior):** This week focuses on making the triage process fully interactive and refining Pepper’s social behavior. **Amr** and **Amira** collaborate to transform the triage questionnaire into a **conversation**. Amr expands triage.py to include a list of questions Pepper should ask and logic to interpret yes/no answers. Amira helps by integrating this with Pepper’s dialogue: for example, Pepper will ask the questions one by one (using TTS to speak them) and either listen for a spoken “yes”/“no” or accept input via the tablet buttons. They implement logic so that after each answer, Pepper (or the UI) moves to the next question until completion, then computes the score. **Aly** programs Pepper’s **gestures and expressions** to make interactions more natural: Pepper might nod after the patient answers, or use hand gestures when speaking. He also ensures Pepper’s navigation is considered – for instance, during triage, Pepper should remain facing the patient; if Pepper had moved around, he makes sure Pepper can localize the user or stay in place for the conversation. Additionally, Aly tests a “navigation demo”: Pepper autonomously moving a short distance (like guiding a user to a room or simply moving towards a designated point) to lay groundwork if they want to show this in the final presentation or during hospital deployment (like escorting patients). **Mohammed** refines the UI for triage: possibly implementing a dedicated triage screen that highlights each question as Pepper asks it, and shows a summary at the end (“Your responses have been recorded”). He also ensures that the UI and Pepper’s spoken prompts stay synchronized (e.g., if the patient taps “Yes” on screen, Pepper hears it or at least the system registers it without Pepper also expecting a voice response). **Abdulrahman** works under the hood to manage the state of the conversation: he updates the backend or Pepper’s logic so that it knows when it’s in “triage mode” vs “FAQ mode” vs idle, making sure Pepper doesn’t accidentally answer an FAQ in the middle of a triage dialog or vice versa. **Youssef** assists with testing this interactive triage: acting as a patient, he goes through the triage Q&A flow multiple times (with different yes/no patterns) and checks that the scoring and outcomes are correct and that Pepper’s responses are appropriate (e.g., Pepper thanks the patient after completing the triage). By the end of Week 12, the triage interaction should feel more like a natural back-and-forth (even if via button presses) rather than a static form, and Pepper’s behavior is more lifelike and aligned with the conversation.
* **Week 13 (Nurse Alert Integration):** Now the team tackles the nurse alert feature in full. **Mohammed** and **Youssef** implement the connection to the **Andalusia Hospital’s nurse alerting system or app**. If an official API is available (for example, to create an alert in the hospital’s system or send a push notification to the Andalusia mobile app), Mohammed writes the integration in the backend (e.g., an API call or database entry when a high-severity triage is submitted). If no existing system can be tapped into, they create a simple workaround: possibly a dedicated “Nurse Alert” Android app or a web dashboard that listens for alerts from the Pepper backend. Youssef might write a small script or use a service like Firebase Cloud Messaging to send a notification to a test nurse’s phone when triggered. On Pepper’s side, **Aly** and **Abdulrahman** extend the Pepper alert behavior: instead of just saying an alert phrase, Pepper could now also display a red alert message on the tablet (“**Nurse alerted: Urgent assistance needed**”) for clear visibility. They test the full chain: a patient goes through triage, the triage score is critical, the backend triggers an alert – Pepper announces the urgent case *and* a nurse’s phone (or a screen dashboard) receives a notification about the case. **Amr** refines the threshold for what constitutes an “urgent” case to minimize false alarms (he may incorporate some simple logic like if triage score above a certain number triggers alert). **Amira** ensures that if the patient asks a question outside Pepper’s scripted flow (like suddenly asking an unrelated FAQ during triage), Pepper can handle it or gently steer back – this might be fine-tuning but helps Pepper appear more coherent. The end of Week 13 should see a **working nurse alert system**: both Pepper’s on-site alert and an off-site notification are integrated and tested in a demo scenario. This is a key integration milestone, as now all major components (voice FAQ, triage, alert) are connected.
* **Week 14 (System Refinement & Buffer):** This week is kept as a buffer to finish any pending integration and start polishing the system. The team collectively tests the **entire pipeline** multiple times. **Abdulrahman** focuses on backend robustness: adding error handling and timeouts (e.g., if Whisper takes too long or fails to transcribe, the system should handle it gracefully by asking the user to repeat). He also ensures that the system can handle multiple users sequentially – for instance, once one patient’s session is done, Pepper can reset for the next interaction without needing a reboot. **Mohammed** polishes the UI based on integration tests: maybe adding a restart or back button, clarifying any Arabic translations, and ensuring consistency (all messages Pepper speaks are also shown on screen for accessibility). **Aly** double-checks Pepper’s hardware readiness for sustained use: he calibrates Pepper’s sensors if needed, cleans up the robot’s code (removing any debug movements that aren’t needed), and ensures Pepper’s battery is managed (perhaps plan to have Pepper on charger between sessions or identify how long Pepper can run during testing). **Amira** might work on expanding the chatbot’s capabilities slightly – for example, adding a few more fallback answers for unrecognized questions (“I’m sorry, I didn’t catch that. Could you rephrase or ask a nurse for help?”) to improve user experience. She could also integrate a small talk or greeting skill if time permits (Pepper saying “Hello, how can I assist you today?” proactively). **Amr** uses this buffer week to run a pseudo-*evaluation* of the triage model: he might simulate 10 hypothetical patients with different symptoms to see if the triage scoring aligns with what a medical professional would expect. If he finds any discrepancies (like the model underestimates a clearly urgent case), he adjusts the rules or retrains the model if using ML. **Youssef** consolidates documentation for each module (important as the project grows complex). He makes sure that the user manual (for hospital staff) and technical docs (for the team/instructors) are up to date, describing how to start the system, how each scenario works, and how the nurse alert is triggered. By the end of Week 14, the team should have tied up loose ends and be entering the final phase with a mostly complete system.
* **Week 15 (Internal Testing & Quality Assurance):** The team conducts thorough internal testing this week, essentially a simulated **alpha test** of the system. They set up Pepper in a controlled environment and run through **multiple patient scenarios** from start to finish, as if in a hospital. Scenarios include: a simple check-in and no further action, a patient asking several FAQs in a row, a patient going through triage with non-urgent answers, and one with urgent answers triggering the nurse alert. **Youssef** leads the testing effort, recording the outcome of each scenario and any issues encountered. **Aly** monitors Pepper’s performance – checking if Pepper’s movements or sensors cause any trouble (for instance, does Pepper hear the patient well in the presence of background noise? Does it maintain proper body orientation toward the user?). **Amira** and **Abdulrahman** focus on the dialogue and backend performance – verifying that speech recognition accuracy is acceptable and that responses come quickly (if they notice delays, they might consider optimizations like using faster Whisper models or simplifying the pipeline). **Mohammed** checks the UI’s reliability – no crashes or unresponsive buttons after extended use – and that the tablet always stays synced (for example, Pepper’s tablet doesn’t accidentally revert to a default screen or sleep during interaction; he might adjust tablet settings to prevent sleep). **Amr** examines the triage outcomes from these tests to ensure consistency – if a particular combination of answers gave an unexpected score, he debugs and fixes the logic. By the end of Week 15, the team compiles a list of any **bugs or improvements** needed before deploying in a real environment. They fix critical bugs immediately (e.g., if Pepper missed recognizing “no” in Arabic, add that to the vocabulary or adjust mic sensitivity) and schedule minor enhancements for next week. The outcome is a verified, quality-assured system ready for pilot testing.
* **Week 16 (Usability Improvements & Pilot Prep):** In the final week of this phase, the team focuses on usability tweaks and preparing for a small pilot at the hospital. **Mohammed** implements any last-minute UI enhancements identified during testing – for instance, if users seemed confused at any point, add guidance on the screen (“Please speak your question now” prompt or similar). He also ensures the UI is branded appropriately (perhaps adding the hospital logo or colors, since this will be seen by hospital stakeholders during the pilot). **Amira** fine-tunes Pepper’s verbal responses for politeness and clarity, possibly adding variation to make Pepper sound more natural (like adding “please” and “thank you” where appropriate). **Aly** sets up a stable deployment configuration: he makes sure Pepper can start all necessary services on boot (so that if Pepper restarts, the bridge and tablet UI automatically reconnect to the backend). He also checks Pepper’s physical condition (tighten any loose joints, etc.) given it might be transported to the hospital. **Abdulrahman** and **Youssef** together run a **security and data privacy check** – since patient data might be involved, they ensure that no sensitive data is stored improperly (for example, they clear any logs that contain personal info, and secure the communication channels with at least a simple authentication or within a closed network). They also verify that running everything locally (without internet) works smoothly, as planned, to comply with privacy. **Amr** prepares a short training document or cheat-sheet for the nurses or hospital staff who will interact with the system during the pilot (e.g., what to do if Pepper flags an urgent case, or how to reset the system if needed). By the end of Week 16, the team is confident that the Pepper assistant is user-friendly and robust, and they are ready to test it in the real setting.

## **Weeks 17–21: Pilot Deployment and System Refinement**

* **Week 17 (Pilot Deployment in Hospital – Phase 1):** **Milestone: First Hospital Trial.** This week, the team deploys the Pepper robot in **Andalusia Hospital** for a supervised pilot test (likely in a controlled area such as the hospital lobby or a clinic waiting area). **Aly** transports and sets up Pepper on-site, ensuring network connectivity to the backend (which might run on a local server or a powerful laptop brought along). The team conducts a live test with hospital staff observing. **Youssef** coordinates test participants: a few hospital employees or volunteer patients interact with Pepper as if checking in for an appointment or seeking information. The focus is on observing real-user behavior. **Amira** and **Abdulrahman** closely monitor the system during these interactions: Amira watches for any speech recognition issues (strong accents, background noise in the hospital might affect Whisper) and Abdulrahman monitors the backend logs for errors or slow points. **Mohammed** gathers direct feedback from users and nurses on the interface – for example, do users know what to do when they see Pepper, do they find the touch-screen helpful, is the Arabic translation accurate, etc. **Amr** pays attention to the triage outcome in a real context; if a test user triggers an urgent alert, he checks that the nurse on duty receives it and how they respond. At the end of the pilot session, the team meets with hospital stakeholders (perhaps the IT or nursing supervisor) to gather their feedback and suggestions. By the end of Week 17, the team will have invaluable data on usability and any environment-specific issues (like Wi-Fi dead zones or Pepper’s voice volume in a busy lobby). They document all findings.
* **Week 18 (System Tweaks After Pilot):** Back to the lab after the pilot, the team addresses the issues discovered. **Abdulrahman** fixes any backend issues that arose (for example, if the system crashed when two users tried to use Pepper back-to-back, he will add concurrency handling; if network latency was an issue, consider localizing more processing on Pepper). **Amira** refines the speech recognition and NLU based on pilot data – she might find that certain Arabic phrases weren’t recognized, so she could incorporate a workaround (like adding those phrases to a small custom dictionary or using Whisper’s translation mode if needed). **Mohammed** adjusts the UI if users seemed confused – for instance, maybe add an initial greeting screen or an obvious “Start” button to engage users, or larger Arabic font if it was hard to read. **Aly** may tweak Pepper’s positioning or motion: if in the hospital Pepper needed to turn more to face people or if floor slippery affects navigation, he accounts for that (maybe limit movement during interactions for stability). **Amr** fine-tunes the triage questions or thresholds if the hospital medical staff had suggestions (e.g., maybe they want an additional question about a symptom that was missing, or a different cutoff for calling a nurse). **Youssef** updates the testing plan for a second pilot or final demo, incorporating what they learned (for example, instructing a staff member to always be on standby during Pepper’s operation in case Pepper mishears and needs assistance). By the end of Week 18, the system is updated to a **Beta version** that is more aligned with the real-world environment.
* **Week 19 (Extended Testing and Reliability):** The team performs another round of extensive testing, this time focusing on reliability and edge cases. **Aly** runs Pepper continuously for several hours to ensure the system can handle extended usage (monitoring for any memory leaks or overheating on Pepper). He also tests how Pepper recovers from network outages – e.g., disconnect Wi-Fi for a moment and see if Pepper rejoins smoothly. **Abdulrahman** simulates high load or rapid-fire interactions to test the backend’s stability (though in reality Pepper will handle one user at a time, they ensure the system won’t break if one user quickly asks many questions). **Amira** tests edge speech cases – like someone speaking very softly, or two people talking to Pepper at once; while the system might not fully handle overlapping speakers, they decide how Pepper should behave (perhaps Pepper detects multiple voices and says “Please speak one at a time”). **Mohammed** and **Youssef** perform a quick **accessibility review**: consider users who are hard of hearing or non-native speakers. Pepper already provides answers in text and voice; they confirm the text contrast and size are sufficient, and maybe add an icon-based guidance if useful (like a microphone icon for voice, etc.). **Amr** stress-tests the triage logic with unlikely combinations to ensure it never produces an error (like if somehow all answers are blank or a question is skipped, the code should handle it). By end of Week 19, the team has ironed out rare issues, aiming for a system that can run all day in the hospital without crashing and can handle the variability in human interaction.
* **Week 20 (Final Pilot & Usability Evaluation):** The team schedules another trial at Andalusia Hospital – this time effectively a **final rehearsal in the real setting** and an opportunity to formally evaluate usability. **Youssef** prepares survey forms or interview questions to gather user satisfaction metrics from those who interact with Pepper. The robot is placed again in the hospital for a day (or several hours) to interact with real visitors. **Mohammed** and **Amira** observe from the side, noting how intuitively people approach Pepper and whether they follow Pepper’s prompts. Is the bilingual support meeting their needs? For instance, do Arabic speakers get the info they need easily? **Abdulrahman** ensures any data (like check-in info or triage results) are being logged securely, which will help evaluate how the system was used. **Aly** checks Pepper’s hardware throughout the day, ensuring the battery is charged and possibly swapping batteries or having Pepper dock if needed. **Amr** communicates with the nursing staff to see if the alerts are helpful and how they manage an alert from Pepper – this helps gauge the practicality of the feature. At the end of this pilot, the team collects all feedback and usability data. For example, they may find that *80% of users were able to get the info they asked for without human help*, or that *some users were shy to speak to a robot*, etc. They compile these findings into a report. By the end of Week 20, they have a clear picture of the system’s usability and any last tweaks needed before the final presentation and handoff.
* **Week 21 (Final Adjustments and Freeze):** In this week, the team implements any *minor* adjustments indicated by the latest pilot’s feedback, but avoids major changes (to keep the system stable). **Abdulrahman** might refine the backend one last time for performance (perhaps caching answers to FAQs to respond faster, or adjusting Whisper model size if latency was noted by users). **Amira** adds any last FAQ answers if new common questions came up during the pilot. **Mohammed** ensures the UI is polished (all text finalized, any placeholder content replaced with real info from the hospital). **Aly** sets the final configuration on Pepper: for example, Pepper’s startup behavior is configured to automatically launch the application so that hospital staff won’t need to manually start anything. He also verifies that Pepper’s volume, language settings, and network are all persistent. **Amr** double-checks the triage and alert threshold with any final input from medical advisors to ensure the system is safe and errs on the side of caution (better to alert a nurse too often than not at all, perhaps). **Youssef** finalizes the project documentation and prepares a slide deck and possibly a video demonstration for the final presentation. He includes results from the usability testing (e.g., user satisfaction rates, any improvements in efficiency observed). By the end of Week 21, the team declares a **“feature freeze”** – the Pepper medical assistant robot is fully implemented and tested. The remainder of the schedule will focus on deployment and presentation.

## **Weeks 22–25: Deployment, Final Testing, and Final Presentation**

* **Week 22 (Full System Deployment at Hospital):** In this week, the team deploys the final version of the system on-site for an extended period (if permitted by the hospital). This could be considered a beta deployment where Pepper is actually used in day-to-day operations in a limited capacity. **Aly** installs the backend server in a reliable location (possibly on hospital premises or a cloud server if allowed, though local is preferred for privacy) and ensures Pepper connects to it over a secure network. Pepper is stationed at, say, the hospital entrance or reception. The team likely alternates presence on-site to supervise but tries to let hospital staff and real patients use Pepper naturally. **Abdulrahman** and **Mohammed** might train a few hospital staff or IT personnel on how to start/stop the system and basic troubleshooting (documenting this in a user manual). **Youssef** continues to gather usage data (number of interactions, any errors) to gauge reliability in a real deployment. If any critical issue is found (e.g., Pepper froze after many uses), the team addresses it rapidly. By the end of Week 22, Pepper has been essentially **launched in a real environment**, and the team prepares to wrap up the project.
* **Week 23 (Monitoring and Support):** The system has been deployed, so this week the team acts as support and monitors its performance in the hospital. They likely reduce on-site presence, intervening only if something goes wrong, which tests the **independence** of the system. **Aly** remains on call for any robot hardware issues (e.g., if Pepper needs a reboot or maintenance). **Amira** and **Abdulrahman** monitor logs remotely (if possible) to ensure speech queries and backend requests are functioning without errors. **Mohammed** stays in contact with hospital staff to get qualitative feedback – are patients happy using Pepper? Do nurses find the alert useful or disruptive? **Amr** double-checks that no false urgent alerts have been triggered in a real setting; if so, he documents the circumstances. **Youssef** compiles all remaining feedback and performance data in preparation for the final project report. Essentially, Week 23 is about **ensuring stability** over time and making very minor tuning adjustments if absolutely necessary. At this point, the project’s technical work is essentially done, barring any surprise issues.
* **Week 24 (Final Presentation Prep and Documentation):** The team shifts focus to the final presentation and project handoff. **Youssef** leads the effort in writing the final report and preparing presentation materials. This includes summarizing the project’s objectives, design, implementation, and results (citing how it was tested at Andalusia Hospital and the outcomes). **Aly** prepares a live demo plan for the final presentation – depending on logistics, they might bring Pepper in to demonstrate live to the evaluation committee, or if not feasible, have a recorded video of Pepper in action at the hospital. **Amira** gathers evidence of the NLU working (maybe showcasing some example interactions logs or a quick live Q&A with Pepper during the presentation). **Amr** prepares to present the triage component results – possibly with statistics like “Out of X test cases, our triage system matched nurse assessments Y% of the time” if they have such data, or simply explain how it ensures urgent cases get flagged. **Mohammed** compiles user feedback and any UI before-and-after snapshots to show how the interface improved through the project. **Abdulrahman** might prepare a system diagram poster or slide to explain the architecture clearly one more time (Pepper <-> Backend <-> Tablet, and external app) for the final presentation. The team also makes sure to acknowledge Andalusia Hospital’s role and how the system could be expanded or maintained after the project. They rehearse the final presentation, with each member’s speaking part well-timed and the demo (live or video) cued up. By end of Week 24, the documentation and presentation are ready.
* **Week 25 (Final Evaluation and Project Handover):** **Milestone: Final Presentation and Demo.** The team delivers the final presentation to the examiners, faculty, and possibly hospital representatives. They demonstrate the Pepper Medical Assistance Robot in its final form: Pepper greets in Arabic or English, handles check-ins, answers a range of FAQ in both languages, and guides a user through a health triage quiz, ultimately triggering a nurse alert for a simulated critical patient. All features are fully integrated and shown working seamlessly. The presentation highlights include Pepper’s voice-interaction (via Whisper ASR and the RAG-based FAQ system), the intuitive tablet interface on Pepper (powered by Android and web tech), the autonomous aspects (Pepper moving or gesturing appropriately), and the backend intelligence (triage scoring and alert logic) that ensures serious cases get attention. Each team member speaks to their contributions one final time, and the team shares results from the hospital testing – demonstrating the real-world impact and feedback (e.g., *“During our hospital pilot, Pepper was able to answer 90% of visitor questions, and nurses reported the triage alerts helped them respond faster to urgent cases.”*). After the presentation, the team hands over all project materials (code repository, documentation, user manuals) to the university or the sponsor. The project is now complete, with Pepper ready for continued use or further development at the hospital. The 25-week plan concludes with a successful deployment and a confident, comprehensive final presentation, showcasing a Pepper-based hospital assistant that meets the project goals.