HomeQuest

Hyeonseo Shim
Department of Information Systems
Hanyang University
Seoul, Republic of Korea
hyeonseo321@gmail.com

Sihyun Jang
Department of Information Systems
Hanyang University
Seoul, Republic of Korea
jjj99nine2@hanyang.ac.kr

Jin Yun
Department of Information Systems
Hanyang University
Seoul, Republic of Korea
jinijini0402@hanyang.ac.kr

Yeonsoo Cho
Department of Public Administration
Hanyang University
Seoul, Republic of Korea
meriel1010@hanyang.ac.kr

***Abstract*—This study addresses the limitations of current smart home services, which primarily provide data and rely on users to take action on their own, making it difficult to promote sustained behavioral change and interaction within the family.To overcome this issue, we propose a family-centered AI challenge system that uses IoT data to deliver personalized behavioral goals and real-time feedback. The system integrates data from LG ThinQ devices to generate tailored challenges in household chores, energy saving, and wellness, while visualizing each member's progress through a character-based interface that promotes participation. Users earn points by completing challenges and can view family rankings, and the accumulated data enables the AI to analyze daily patterns and offer targeted insights. By transforming everyday data into a game-like, immersive experience, the system enhances engagement and achievement within the household and shifts the smart home from an efficiency-focused model to an interactive environment that supports family connection and positive routines.**

TABLE I. *Role Assignments*

Role	Name	Task Description
Backend Developer	Yeonsoo Cho	The Backend Developer builds and manages server-side architecture, databases, and APIs that support the application's core features. This includes designing efficient data models, implementing security and authentication, and ensuring reliable data exchange between systems. The developer enhances performance, scalability, and fault tolerance while maintaining documentation and monitoring. Through collaboration with frontend and AI teams, the backend developer ensures stable and efficient platform operations.
AI Developer	Hyeonseo Shim	The AI Developer designs and implements intelligent systems that enhance the app's personalization and decision-making capabilities. This involves collecting and preprocessing data, training and fine-tuning machine learning models, and deploying them through scalable APIs or pipelines. The developer evaluates model performance, iterates for improvement, and ensures seamless integration with backend and frontend systems. The AI Developer uses data science and engineering to deliver smarter, adaptive experiences and real-time insights.
Frontend Developer	Sihyun Jang	The UI/UX Designer is responsible for understanding user needs and transforming them into intuitive, user-centered designs. This role involves conducting user research, creating wireframes and interactive prototypes, and developing a consistent design system that aligns with the app's identity. The designer ensures visual harmony, accessibility, and usability across all interfaces while collaborating closely with developers to maintain design accuracy during implementation. By continuously refining layouts and interaction flows, the designer contributes to creating a seamless and engaging overall user experience.
Frontend Developer	Jin Yun	The Frontend Developer builds responsive, interactive, and visually precise user interfaces from design specifications. This includes creating components with modern frameworks, ensuring cross-device compatibility, and integrating backend APIs. The developer improves performance and accessibility, conducts testing and debugging, and maintains clean code. Collaborating with designers and backend teams, the frontend developer delivers a smooth and cohesive user experience from design to deployment.

I. Introduction

A. Background

1) Smart homes have become deeply integrated into our daily lives, offering convenience and automation in many ways. However, most smart home data only shows the results of what has already happened through numbers and notifications. While users can review their activity outcomes, the data rarely helps them recognize or improve their living patterns. In fact, as smart homes provide more information, the gap between data and meaningful behavioral change continues to grow. We wanted to close this gap.

2) This realization naturally led us to focus on the idea of home as a shared space. Daily life is not created by individuals alone, but through the interactions and relationships among family members. Yet current smart home systems

manage data on an individual basis, failing to reflect this interconnectedness. We believe smart homes should evolve into systems that allow families to recognize each other's actions and grow together through shared feedback.

3) From this perspective, we propose HomeQuest. The service analyzes everyday data collected from LG ThinQ devices to generate personalized goals, reports, and real-time feedback tailored to each family's lifestyle patterns. Everyday actions such as finishing the laundry, making small energy saving decisions, or maintaining a regular water drinking habit are recorded and visualized through a character interface and point system, encouraging natural participation and interaction among family members. Through this process, smart home data is transformed from simple information delivery into a gamified experience that motivates families and fosters positive habits, presenting a new direction for smart homes.

B. Problem Statement (Client's Needs)

(a) Lack of Family Connection and Behavioral Change
Current smart home systems are primarily operated through individual user accounts, managing each member's data separately. As a result, even though home appliances are shared devices used by the entire family, the collected data remains divided by individual and fails to connect the activities of different members. This separation makes it difficult to understand collective household patterns or identify shared goals, preventing the data from contributing to meaningful habit improvement or behavioral change. While smart homes effectively enhance personal efficiency and convenience, they show clear limitations in promoting family interaction and collaborative experiences.

(b) Need for Meaningful Data Experience
Current smart home systems, powered by IoT and AI technologies, collect and provide large amounts of data. However, users receive this information repeatedly without meaningful context. Notifications about laundry completion, energy consumption accumulate over time, leading to information overload and increasing user fatigue. This constant flow of data diminishes interest and engagement, causing users to perceive the smart home merely as an automated tool rather than a supportive companion.

(c) Lack of Emotional Connection
The current smart home data environment focuses mainly on storing and displaying information, failing to capture the emotional flow or meaningful changes in daily life. In this fragmented structure, even when family members experience behavioral or lifestyle changes, the emotional meaning behind these actions is not shared, weakening the overall context of family life. While smart homes emphasize functional convenience, they lack mechanisms that foster emotional interaction and connection among family members. As a result, positive changes often fade unnoticed, and the family's growth and shared experiences remain buried within disconnected data.

C. Common Concepts and Object Definition

- Family: A container that groups family members. Each family entity maintains its own set of policies, including reward and penalty rules, cutoff times, and notification preferences.
- Member: An individual family participant who can be linked to one or more LG devices or wearables.
- Goal: A quantitative target defined at either the individual or family level (e.g., "Do laundry twice a week," "Reduce family energy use by 10%"). Each goal includes parameters such as duration (daily, weekly, or monthly), baseline values, measurement units, success criteria, and weighting or contribution rules.
- Challenge: An executable package composed of one or more goals. Challenges can be individual- or family joint-, and include details such as start and end dates, rewards and penalties, and verification methods.
- Metric: A standardized measurement unit used to quantify performance or behavior. Examples include laundry cycles (count), energy consumption (kWh), step count (steps), and cleaning area (m^2).
- Event: A single recorded action triggered by a device, app, or voice command—for instance, completing a laundry cycle, starting or ending a robot vacuum session, pressing a confirmation button, or saying "I'm done."

D. Research on Related Software and Algorithm

(a) Samsung SmartThings Energy [1]

1) Function: Samsung SmartThings Energy automatically collects and analyzes power consumption data from home appliances to help users manage energy usage efficiently. Within the app, weekly and daily consumption patterns and reduction rates are visualized, accompanied by notifications such as "Power usage decreased by 8% this week" and energy-saving tips. In addition, the system includes functions that automatically set energy-saving goals and provide personalized recommendations for more efficient usage.

2) Similarity with Our Service: Similar to our service, SmartThings Energy analyzes lifestyle patterns based on appliance data and presents them in report form. Both services share the same ob-

jective of promoting efficient household behavior through data-driven and personalized feedback mechanisms.

3) Difference from Our Service: While SmartThings Energy focuses primarily on energy efficiency and power consumption management, our service integrates data related to appliances, health, emotions, and lifestyle habits to provide a comprehensive AI-generated family lifestyle report. In other words, it goes beyond a single-domain approach (energy management) to evolve into a holistic AI platform that offers family-oriented lifestyle improvement and emotional feedback, establishing a clear point of differentiation.

(b) Strive (Group Fitness Challenge App) [2]

1) Function: Strive is a social fitness challenge application that allows users to set exercise goals and participate in team-based competitions or collaborations with family members or friends. Users can set objectives such as step counts or workout durations and monitor their progress through a real-time leaderboard. Upon completing missions, users earn badges and titles, and family members can encourage one another through motivational notifications. Additionally, the app enables continuous performance evaluation by comparing individual results with team averages.

2) Similarity with Our Service: Like the Family Challenge System in our service, Strive adopts a team-based goal structure and promotes participation through both competitive and cooperative mechanics. Both platforms use behavioral data to generate visual feedback and employ gamification strategies that enhance engagement and motivation.

3) Difference from Our Service: Strive is limited to fitness-related data derived from wearable devices, whereas our service expands the scope by combining appliance, health, and lifestyle data to support challenges across various daily activities such as cleaning and energy saving. Thus, instead of functioning merely as a fitness-oriented competition platform, our system aims to build an intelligent challenge ecosystem that integrates family communication and overall lifestyle improvement within the household context.

(c) FamilyAlbum (by Mixi, Japan) [3]

1) Function: FamilyAlbum is a private family-oriented social networking platform that allows members to share photos and videos and interact through comments and emojis. The AI system automatically categorizes uploaded content by date and generates features such as "1-second highlight videos" or "On this day last year," helping families relive shared memories. Furthermore, its feed-based interface facilitates easy intergenerational communication and emotional connection.

2) Similarity with Our Service: Our service also employs a family-centered feed structure and AI-based summarization functions to promote communication and emotional interaction among members. Like FamilyAlbum, it visualizes daily records in an easily accessible and aesthetically organized format, offering a similar user experience focused on familial engagement.

3) Difference from Our Service: While FamilyAlbum primarily emphasizes emotional sharing through photo and video records, our service goes a step further by transforming AI-collected lifestyle data into narrative-driven visual stories within the family feed. This distinction positions our service not merely as a record-oriented platform but as an AI-driven family relationship enhancement system, combining emotional exchange with data insights to create a new form of intelligent family communication platform.

(d) Reinforcement Learning [4]

1) Function: Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns to make optimal decisions through continuous interaction with its environment. By receiving rewards or penalties in response to its actions, the agent gradually develops a policy that maximizes cumulative long-term rewards. Through repeated trial-and-error processes, the system is able to discover effective strategies even without prior knowledge of the environment's structure.

2) Application to our Service: In the proposed family challenge service, RL can be applied to the challenge recommendation and difficulty adjustment algorithm. The system observes user engagement metrics—such as completion rates, satisfaction feedback, and participation frequency—and interprets them as reward signals. A successfully completed and positively rated challenge yields a high reward, whereas prematurely abandoned or negatively rated challenges result in low rewards. Based on these reward patterns, the RL agent updates its policy to recommend future challenges that are more likely to match the family's interests and ability levels.
Furthermore, RL enables dynamic difficulty adjustment, where overly easy or overly difficult challenges are penalized, while moderately challenging and successfully completed ones are rewarded.

This allows the system to automatically tailor the level of difficulty to each family's performance patterns.

3) Expected Outcomes: By integrating reinforcement learning, the service can (1) continuously personalize challenges to fit each family's behavioral tendencies, (2) increase long-term engagement and motivation, and (3) evolve into a self-adaptive recommendation system that improves its performance as more interaction data is accumulated.

II. Requirements

A. User Authentication

a) Login Screen Initialization
- The application opens with the login screen as the default interface.
- Returning users can sign in, while new users may register or recover their credentials.
- The password field is masked to ensure privacy.

b) User Validation
- Users enter their email and password into the provided fields.
- Email format is checked for validity before submission.
- Password must satisfy Firebase Authentication's minimum security criteria.

c) Authentication Process
- Login requests are processed through Firebase Authentication.
- Upon successful authentication, Firebase returns a unique user identifier (UID).
- The application retrieves additional user data (family ID, nickname, role, participation status) from Firestore.
- Retrieved data is stored locally (Zustand) for in-app use.

d) Post-Login Routing
- After authentication, users are redirected to the main tab-based interface.
- Users without an assigned family are guided into the onboarding flow (family creation or joining).

B. Onboarding Flow

a) Profile Setup
- Users enter a nickname, select their family role (e.g., parent, sibling), and specify their residence.
- The system validates nickname availability through a duplication check.
- The system stores the user's nickname, family role, and residence information for profile initialization.

b) Avatar Creation
- Users create their personal avatar by selecting one of the predefined characters.
- Avatar selection is saved as part of the user's profile configuration.

c) Family Connection
- Users can join an existing family by entering a family code.
- Users may alternatively create a new family if no existing group is available.
- The system stores the final family association and grants access to the main interface after setup is complete.

C. Tap1. Home Screen

a) Home Appliance Map
- A house-shaped map with multiple rooms appears when the user enters the main interface.
- Each family member is placed in their own room, and their character is displayed inside that space.
- All registered LG appliances appear in the appropriate rooms with identifiable LG icons.
- Users can zoom in and navigate around the map to explore different rooms.
- A button in the bottom-left corner allows users to register new LG appliances.
- A button in the bottom-right corner opens Today's Report, showing challenge results, AI-based pattern analysis, and earned points.

b) Character Detail
- Users can tap a character to view that member's current status in real time.
- The screen displays the member's in-progress challenge.
- Recently completed challenges are shown as collectible cards.

D. Tap2. Challenge Screen

a) Challenge Overview
- Users can view challenge filters such as “Me”, “Family”, and “All”.
- Category filters (e.g., Energy Saving, Household Tasks, Wellness) help users narrow down challenge types.
- A summary section displays the user’s total number of participated missions, completed missions, and overall success rate.

b) In-Progress Challenges
- The screen lists all challenges the user is currently participating in.
- Each challenge card shows progress, mission type, and remaining time or remaining actions.
- Users can tap a challenge card to open the detailed challenge screen.

c) Recommended Challenges
- The system suggests new challenges based on the user’s family activity patterns.
- Each recommended challenge card displays the estimated reward points and the required activity.
- Users can tap the “Start” button on the card to immediately join the selected challenge.

d) Challenge Detail Screen
- Users can view the full challenge description, including mission flow, participants, and reward structure.
- Progress visualization is provided through a step-based or bar-based progress indicator.
- Comment and reaction features allow family members to communicate within the challenge.
- Users can add new comments, reply to others, and react with heart icons.

E. Tap3. Reward & Market

a) Personal Reward Overview
- Users can view their current personal reward points at the top of the Reward tab.
- A message below the point total shows how many additional points can be earned from ongoing challenges.
- Tapping the personal reward box opens a detailed timeline of point history.

b) Family Reward Overview
- The Reward tab displays the total accumulated family points in a dedicated card.
- A horizontal progress bar shows relative contributions of each family member.
- A weekly family ranking section lists the top contributors for the current week or month.
- Tapping the Family reward box opens a detailed timeline of point history.

c) Point Timeline
- The timeline displays all point gains and deductions in chronological order.
- Each entry includes date, activity name (e.g., Running 3km, Relay Laundry), and the exact point change (+ or –).
- Positive and negative point events are visually distinguished through color-coded labels.
- Users can scroll through past point activity to track their progress over time.

d) Reward Market
- Users can switch to the Market tab to browse available discount coupons.
- A search bar allows users to filter coupons by keyword or brand.
- Each coupon is displayed as a reward card showing required points, reward description, and preview image.
- Users can tap a card to open a detailed view with redemption button.

e) Reward Redemption
- Rewards can be redeemed only when the user has enough points.
- After redemption, the required points are deducted from personal or family balance accordingly.
- A confirmation message appears after successful redemption.

F. Tap4. Family Ranking

a) Regional Family Leaderboard
- Users can browse rankings for families located in the same neighborhood.
- Each family is displayed with its name, point total, and indicators for ranking changes (up, down, or unchanged).

- The top three families appear as medal cards (gold, silver, bronze) for quick recognition.

b) Ranking-Based Rewards
- When a family's ranking increases, the reward box probability increases accordingly.
- Users may receive a reward box containing random gifts such as discount coupons or bonus points.
- When the gauge becomes fully filled, the reward box automatically opens.

c) Seasonal Ranking Information
- A help button opens a popup explaining how the ranking system works.

G. Family Challenge System

a) Challenge Creation and Recommendation
- HomeQuest provides action-oriented challenges that do not require users to manually enter numerical targets.
- Three main challenge domains are supported: Household Tasks, Energy Saving, and Exercise & Wellness.
- Challenge recommendations are generated by analyzing recent individual or family activity patterns.
- Each recommended challenge includes goal intensity (Easy / Normal / Hard) and duration.
- To generate a final recommendation, the system first ranks all available challenges using the predicted completion likelihood and a frequency-adjusted score. From this ranking, the system extracts the top-K most promising candidates. A softmax sampling strategy is then applied to the adjusted scores to stochastically select one challenge. This ensures that high-quality recommendations remain prioritized while still allowing exploration of alternative options.
-

b) Challenge Types

1) Daily Challenge
- An individual mission that must be completed within the same day.
- Grants +10 personal points upon successful completion.
- Automatically refreshes every morning.

2) Speed Challenge
- Only the first person to press the challenge button within the limited time earns the right to perform the mission.
- Grants +20 personal points upon successful completion.

3) Family Relay Challenge
- A sequential mission in which family members complete tasks in a fixed order.
- Failure by any member results in overall failure.
- Grants +50 family points upon successful completion.

4) Family Accumulation Challenge
- All members contribute to reaching a combined target value.
- Grants +40 family points when the shared target is achieved.

c) Challenge Execution and Evaluation
- Users can browse the "Recommended Challenges" section, select a challenge card, and start the mission immediately.
- Challenge evaluation is processed automatically on a daily or weekly basis depending on the challenge type.
- For family-level challenges, all member contributions are aggregated to compute overall progress.

d) Calculation Model
- The system uses an Aggregate Model for all family-level challenges.
- Energy-based missions aggregate total watt-hours (Wh) used during the challenge period.
- Count- or activity-based missions aggregate all measurable values (e.g., steps, chore events) across participating members.
- All members are weighted equally; missing data is replaced with the family average and marked as "Incomplete."
- A challenge is considered successful when the final aggregated value meets or exceeds the target threshold.

H. Data Collection from Devices

The system integrates and manages data collected from various IoT appliances and wearable devices. Each dataset serves as the foundation for analyzing user lifestyle patterns and generating behavioral indicators at the family level.

(a) Household Tasks Data Collection
The HomeQuest system integrates and manages data collected from various IoT appliances and wearable devices. All device data flows into the backend through LG ThinQ webhooks and is normalized before being stored in Firestore.

- Washing Machine: Each completed washing cycle is recorded as a single household task event. Cycle start and end times are captured to track overall laundry frequency.
- Dryer: Every completed drying cycle is logged in the same manner, allowing monitoring of laundry-related activity frequency.
- Dishwasher: Each dishwashing operation is registered as a completed task. These records are used to measure participation and contribution in kitchen-related chores.
- Robot Vacuum: Both the number of cleaning sessions and the total cleaned area are recorded. These metrics reflect not only frequency of use but also contribution to household cleanliness.

If data from any device are missing or incomplete, the system flags the inconsistency and notifies the user to ensure accuracy and transparency of the calculated indicators.

(b) Energy Consumption Data
The system periodically collects and stores power consumption data from LG ThinQ appliances. This information is used to analyze energy-saving performance relative to a baseline or user-defined family goal.

- Refrigerator: Continuous power usage is tracked to monitor efficiency and detect unnecessary consumption.
- Air Conditioner: Operating modes, usage duration, and hourly power draw are collected to evaluate peak-time usage and the effectiveness of energy-saving modes.
- Washing Machine: Energy consumption per washing cycle is measured in kilowatt-hours (kWh), allowing comparison between normal and eco modes.
- TV: Power consumption is logged during both active and standby states to identify idle usage patterns such as nightly standby power.

If any energy data are delayed or unavailable, the system identifies the gap and computes savings indicators only from verified data sources.

(c) Exercise and Wellness Data
The system gathers health and activity information from wearable devices such as LG Health or compatible smartwatches. Collected data are synchronized daily or weekly to generate wellness indicators.

- Step Count: The total number of steps taken per day is recorded as the baseline metric for physical-activity tracking.
- Heart Rate: Continuous or periodic heart-rate measurements are logged to monitor exercise intensity and overall wellness, while abnormal spikes or missing data are filtered out to maintain accuracy.
- Active Minutes: The duration of light, moderate, and vigorous activity is aggregated to represent both the quality and sustainability of physical exercise.

In cases of synchronization failure or incomplete records, the system highlights the issue and alerts the user, ensuring transparency and reliability of wellness indicators.

III. DEVELOPMENT ENVIRONMENT

A. Platform Configuration and Selection Rationale

HomeQuest is designed as a web application based on Firebase Hosting, utilizing key services of Google Firebase such as Authentication, Cloud Firestore, Hosting, and Cloud Functions. Through this integrated architecture, user authentication, data management, server execution, and deployment are all handled within a single platform. Firebase fully supports HomeQuest's real-time data synchronization and family feedback structure, enabling rapid prototyping and significantly reducing the overall development cycle.

- *Integrated Backend Environment:* Firebase allows user authentication, data management, and deployment to be handled in a single unified environment, eliminating the need for separate server infrastructure.
- *Real-Time Data Synchronization:* By leveraging Firestore's real-time listener feature, challenge progress, point updates, and reward exchanges among family members are synchronized across all devices within one second.
- *Cloud-Based Stability and Scalability:* Built upon Google Cloud infrastructure, Firebase ensures stable data processing and transmission even as the number of IoT and wearable devices increases, without performance degradation.

Overall, Firebase was selected as the optimal platform to achieve HomeQuest's goals of real-time collaboration and consistent family-level data management.

B. Programming Languages and Frameworks

- *Frontend:* JavaScript / React.js Offers high compatibility with the Firebase SDK and provides efficient state

management for real-time data reflection.

- *Backend:* Node.js (Firebase Cloud Functions) Enables serverless function-level execution without the need for traditional server management.
- *AI Module:* Python Used to develop recommendation models and behavior pattern analysis based on family activity data.

C. Database Structure Selection (NoSQL / Firestore)

HomeQuest is a smart home platform centered on real-time data sharing and challenge-based family interaction. To realize this concept, Google Firebase's Cloud Firestore was selected as the main database. Firestore, built on a NoSQL structure, provides flexibility in data modeling and high-speed synchronization capabilities.

- *Flexible Data Structure:* Data entities such as families, members, and challenges are subject to frequent modification. Firestore's schema-free structure allows flexible and scalable data modeling to accommodate these changes efficiently.
- *Event-Oriented Architecture:* HomeQuest handles event-based data such as "family progress status" and "individual completion logs." Storing data in document-oriented structures rather than relational tables enables faster and more efficient query performance.
- *Real-Time Synchronization:* Firestore's real-time listener feature automatically updates all client interfaces when data changes occur. For example, when a family member completes a challenge, the global family leaderboard and character points are immediately updated across all devices.
- *Security Rule-Based Access Control:* Firestore employs rule-based access control, allowing user- and family-level permission management. By utilizing Firebase Security Rules, family data is securely isolated to prevent unauthorized access.

D. Cost Estimation

- Firebase (Spark Plan): Authentication, Firestore, and Hosting services – Free for small-scale usage.
- Development Tools: Visual Studio Code, GitHub, Node.js – Free and open source.
- Domain (Optional): homequest.dev – Approximately ₩10,000–₩15,000 per year.

Firebase's free Spark Plan provides authentication, database, and hosting services at no cost during the development and testing stages.

In the production stage, a **pay-as-you-go** billing system applies depending on usage volume (database reads/writes, storage, and traffic).

E. Development Environment

- Operating System: Windows 11 / macOS Sonoma
- Node.js: v20.x
- npm: v10.x
- Firebase CLI: v13.x
- Integrated Development Environment (IDE): Visual Studio Code
- Browser: Latest version of Google Chrome
- Version Control: Git + GitHub
- Deployment Environment: Firebase Hosting (Google Cloud Infrastructure)

F. Use of Commercial Cloud Platform

HomeQuest utilizes Google Cloud–based Firebase to manage the server, database, authentication, and hosting within a single cloud ecosystem.

Firebase's serverless architecture and auto-scaling capability ensure stable performance even under heavy load or increased IoT device connections.

Moreover, Firebase inherits Google Cloud's TLS encryption and security infrastructure, ensuring that IoT and wearable data are transmitted and stored securely.

This cloud-based structure technically supports HomeQuest's philosophy of real-time data sharing and emotional connection among family members, while maintaining high standards of system reliability, scalability, and data security.

IV. SPECIFICATION

A. Initial Screen

Fig. 1. Initial Screen

When the app is launched, the service logo “HomeQuest” is displayed at the center of the screen. This serves as a welcome and identity screen, allowing users to recognize the brand before entering the main interface.

B. Log In

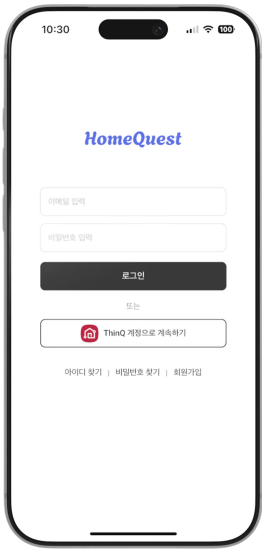


Fig. 2. Login

1) Email Input
   - Description: Prompts users to enter their email address.
   - Input Format: Accepts only valid email formats such as `user@gmail.com`.
   - Validation: Verifies whether the input follows a valid email address format.

2) Password Input
   - Description: Prompts users to enter their password.
   - Input Format: Characters are hidden and displayed as “•” symbols for privacy.
   - Validation: Checks basic rules such as minimum length and non-empty input.

3) Log In Button
   - Action: Attempts to authenticate with the provided email and password.
   - On Success:
     - If the account has logged in before: redirects to the Home screen.
     - If this is the user’s first login: redirects to the Onboarding flow.
   - On Failure: Displays an error message such as “Invalid email or password.”

4) Additional Options
   - Menu: Includes Find ID, Find Password, and Sign Up.
   - External Login: Provides an LG ThinQ login button.

C. Sign-Up

1) Email Input
   - Description: Users enter the local part of their email and select a domain through a dropdown menu.
   - Input Format: local-part @gmail.com
   - Validation:
     - Checks that the local part is non-empty and contains valid characters.
     - Ensures a domain is selected from the dropdown.
     - Verifies full email format before allowing submission.

2) Password Input
   - Description: Users set a password and re-enter it for confirmation.
   - Input Format: Minimum 6 characters.

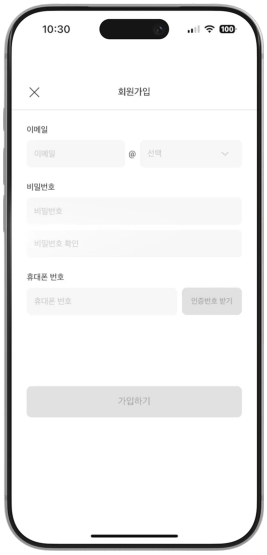


Fig. 3. Sign-Up

Fig. 4. Profile Setup

- Validation:
  - Confirms both password fields match.
  - Ensures the password meets minimum security requirements.

3) Phone Number Input
   - Description: Used for identity verification during sign-up.
   - Input Format: Numeric format such as 01012345678.
   - Verification Flow:
     - Users tap "Send Code" to receive a verification SMS.
     - A verification code field becomes active.
     - Users enter the code to complete phone verification.

4) Sign-Up Button
   - Action: Enabled only when all inputs (email, password, phone verification) are valid.
   - On Success: Completes registration and redirects the user to the Log In screen.

D. Initial Setup

1) Profile Setup
   - Description: This step collects the user's basic profile information after sign-up.
   - Input Fields: Nickname, Family Role, Family Residence
   - Input Format:
     - Nickname: characters using Korean, English letters, or numbers.
     - Family Role: Selected from predefined role options (e.g., Mom, Dad, Sister, Brother, Grandma, Grandpa).
     - Family Residence: use the "Get Current Location" button.
   - Validation:
     - Nickname duplication must be checked through the "Check Duplication" button.
     - "Next" button remains disabled until all fields (nickname, role, residence) are completed.

2) Avatar Creation
   - Description: Users select a 3D avatar that represents themselves.
   - Feature: A horizontal carousel allows users to swipe left or right to browse available avatars.
   - Action: After selecting an avatar, clicking the "Next" button saves the selection and moves to the final step.

Fig. 5. Avatar Creation

3) Family Connection

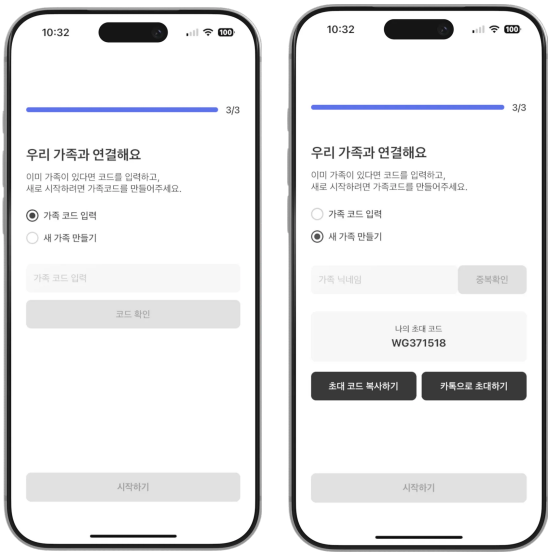


Fig. 6. Family Connection

- Description: Users can either join an existing family group or create a new one.
- Feature Options:
  - Enter Family Code: Users join a family by entering an existing family code.
  - Create New Family:
    - A unique family code is automatically generated.
    - The code can be shared via the "Copy Code" button or the "Invite via KakaoTalk" button.
- Action: After joining or creating a family, clicking "Start" finalizes onboarding and takes the user to the Home screen.

E. Home Tab

Fig. 7. Main Page

1) Home Appliance Map
   - Description: When users enter the main interface, a house-shaped isometric map with multiple rooms is displayed. Each family member occupies their own designated room, and their character is shown inside that space.
   - Components:
     - Isometric House Map: Multi-room layout representing the household structure
     - Character Avatars: One character placed in each room
     - LG Appliance Icons: Registered LG devices displayed in appropriate rooms with identifiable ThinQ-style icons
     - Navigation Controls: Pinch-to-zoom and drag gestures for map exploration
   - Actions:
     - Users can zoom in and navigate across the map to examine rooms in detail.
     - Tapping the bottom-left "+" button allows users to add LG devices.

2) Character Detail
   - Description: Users can tap any character on the home map to view that family member's current status in real time. The detail page presents the

member's ongoing challenge status and recent activity.

- Components:
  - Character Profile: Avatar and basic information for the selected family member
  - Current Challenge Panel: Displays the challenge currently in progress for that member
  - Completed Challenge Cards: Recently finished missions shown as collectible-style cards
- Actions:
  - Tapping a character on the home screen opens the Character Detail Page.
  - Users can check the member's real-time challenge progress and recent achievements.
  - Completed mission cards provide a visual summary of past activities.

3) Bottom Navigation Bar
- Description: A fixed bottom navigation bar allows users to switch between major screens of the app. Tabs provide quick access to Home, Challenge, Reward, and Ranking pages.
- Components:
  - Home: View the home screen and characters
  - Challenge: Access available challenge lists
  - Reward: Check points and reward history
  - Ranking: Compare user rankings
- Actions:
  - Tapping a tab navigates to the corresponding page.
  - The selected tab is displayed with a filled icon and bold text.
  - Fade or slide animations are applied during transitions.

4) AI Report
- Description: Displays the user's daily and weekly challenge progress through an AI-generated report. The section provides personalized insights based on appliance usage patterns and challenge performance.
- Components:
  - Report Card: Central container summarizing AI feedback and completed challenges
  - Challenge Log: Lists current and completed family and personal missions with corresponding points
  - Confirmation Button: Closes the report and returns to the main dashboard

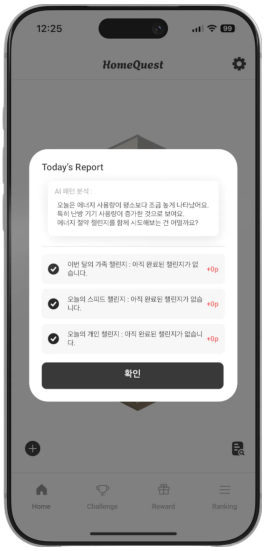


Fig. 8. AI Report

5) Settings Page

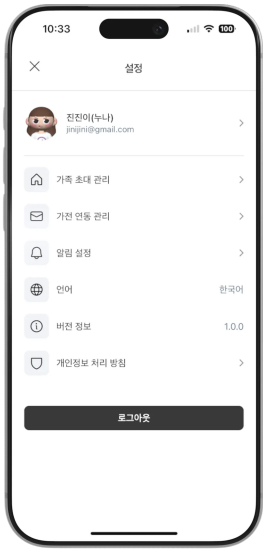


Fig. 9. Settings Page

- Description: Allows users to manage personal information, linked devices, notification preferences, and language settings in a unified interface.
- Components:
  - Profile Header: Displays the user's avatar, name, and email information
  - Settings List: Menu for device connection, family invitation, notifications, and language selection
  - Version and Policy Section: Shows current app

version and privacy policy access

- Logout Button: Provides account sign-out functionality

F. Challenge Tab

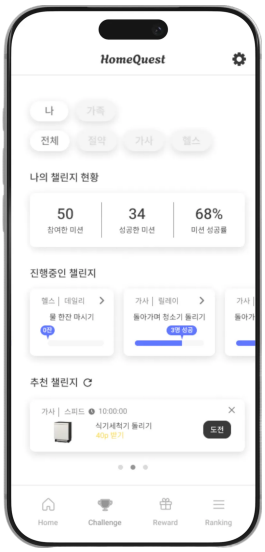


Fig. 10. Challenge Page

1) Ongoing Challenges
   - Description: Displays all currently active challenges in a card-based layout.
   - Components:
     - Category tags (e.g., Health, Energy Saving, Household)
     - Challenge name (e.g., "Walk 10 Minutes," "Laundry Relay Challenge")
     - Progress bar (%) showing completion rate
   - Action: Tapping a card opens the detailed challenge screen.

2) Challenge Detail
   - Description: Shows detailed progress by participant and mission information.
   - Components:
     - Progress timeline (e.g., "Mom completed on 10/1," "Dad completed on 10/4")
     - Duration, difficulty, and reward points
     - Comment section for family encouragement messages with "Like" feature
   - Action:

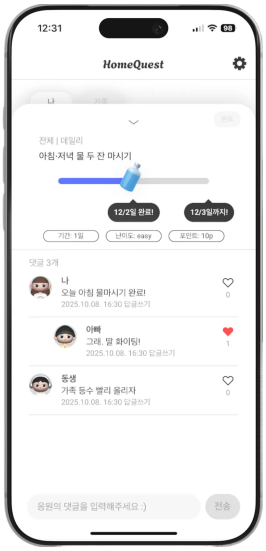


Fig. 11. Challenge Detail

     - Comments are added in real-time when the "Send" button is pressed.
     - Each comment includes username, date, and like count.

3) Recommended Challenges
   - Description: Suggests new challenges based on user activity data.
   - Components:
     - Challenge name, short description, reward points, and "Start" button
     - Example: "Reduce air conditioner use (+20p)," "Do stretching (+5p)"
   - Action: Pressing "Start" adds the challenge to the ongoing list.

G. Reward & Market Tab

1) Reward Overview
   - Description: The Reward tab is the main hub for viewing both personal and family points. At the top, users can switch between the "Reward" and "Market" tabs, with Reward selected by default. From here, users can check their current balance, open the detailed timeline, or move to the Market to spend points.
   - Components:
     - Tab buttons: "Reward", "Market"
     - Personal Reward Card
     - Family Reward Card

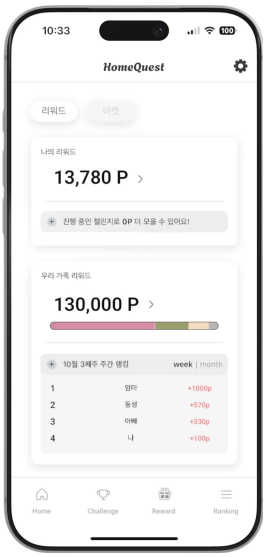


Fig. 12. Reward Page

- Actions:
  - Selecting the “Reward” tab keeps the current summary view active.
  - Selecting the “Market” tab navigates to the Reward Market screen.
  - Tapping the personal reward card “>” button opens the detailed timeline view for that user.

2) Timeline

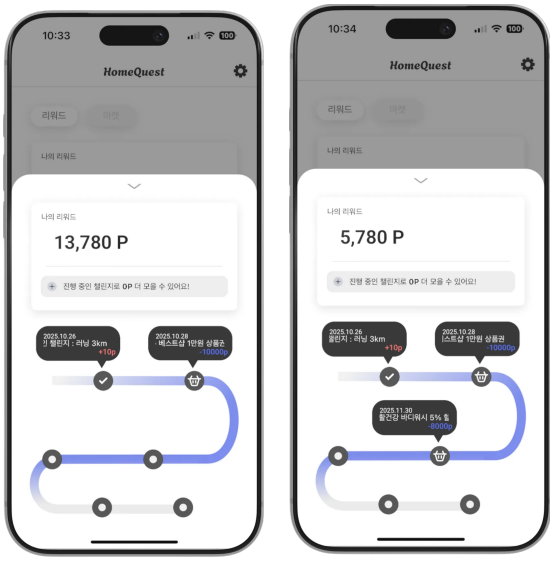
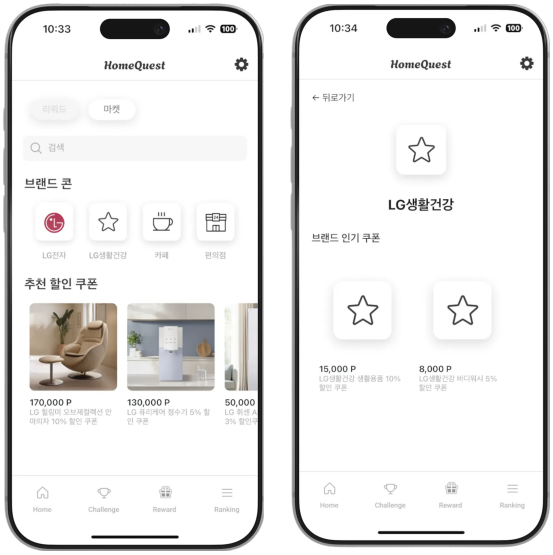


Fig. 13. Reward Timeline

- Description: The Timeline screen shows a chronological history of all point-related events, including points earned from challenges, points consumed in the Market, and other special rewards. Users can understand their entire point lifecycle at a glance.
- Components:
  - Point history list (Challenge name or Coupon name / Date / Point change)
  - Filter buttons: “All / Earned / Used”
  - Period selection dropdown (for example, This Week / This Month / All)
  - Summary bar showing total earned and total used points for the selected period
- Actions:
  - When a coupon purchase is completed in the Market, a new “Used” entry is appended to the list with a negative point value and the coupon name.

3) Reward Market



- Description: The Market screen allows users to exchange their accumulated points for discount coupons and partner benefits. It is accessible by tapping the “Market” tab at the top of the Reward screen.
- Components:
  - Category selector and brand icons (for example, LG Electronics, LG Household and Health Care, cafes, convenience stores)
  - Coupon list for each brand (Image, Name, Required Points, Short description)
  - Coupon Detail Page containing a larger icon, required points, description, and purchase button
- Actions:
  - Selecting a brand shows its popular coupons;

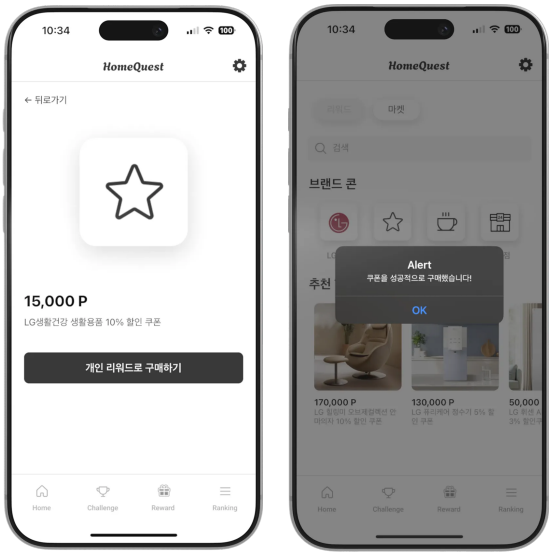


Fig. 14. Reward Market

Fig. 15. Family Ranking Page

tapping a coupon opens the Coupon Detail Page.

- On the Coupon Detail Page, tapping "Purchase with My Reward Points" opens a confirmation dialog.
- When the purchase is confirmed, the required points are deducted from the user's personal balance and a success alert is displayed.
- After closing the alert, the user returns to the Market or Reward screen with the updated balance. At the same time, a new usage record is added to the Timeline.

H. Family Ranking Tab

1) Regional Leaderboard
   - Description: The leaderboard visualizes the point rankings of all participating families in the same neighborhood, highlighting the top performers and the user's own family.
   - Components:
     - Top 3 families displayed with medal icons (Gold, Silver, Bronze).
     - Ranks 1–10 display rank number, family name, points, and up/down arrows.
   - Visual Effects:
     - Rank changes are indicated with arrows (for up, for down).
     - When rising: The row background flashes red for 2 seconds.
     - When falling: The row background flashes blue for 2 seconds.
     - This animation plays automatically during ranking updates, allowing users to perceive changes instantly.
   - Action:
     - Pressing the "?" icon shows a popup explaining the ranking system.
     - Popup text: "Rankings reset on the 1st of every month. Families compete within the same area. Higher ranks increase reward box odds, which include coupons and bonus points."

2) Reward Box
   - Description: Provides visual feedback and rewards for seasonal achievements or ranking milestones.
   - Action:
     - Displays an animated reward box with a message: "Congratulations! +100p."
     - When the user presses "OK," points are instantly added to their account.

V. ARCHITECTURE DESIGN

A. Overall Architecture

The overall software architecture consists of three parts: frontend, backend and AI.

1) Frontend
   - Framework: React Native (Expo)

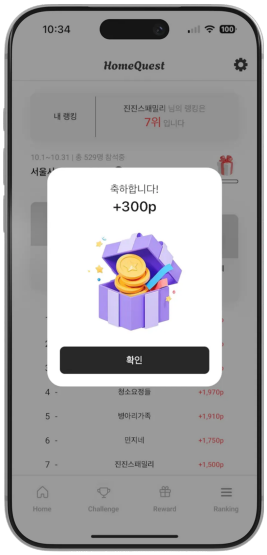


Fig. 16. Reward Box

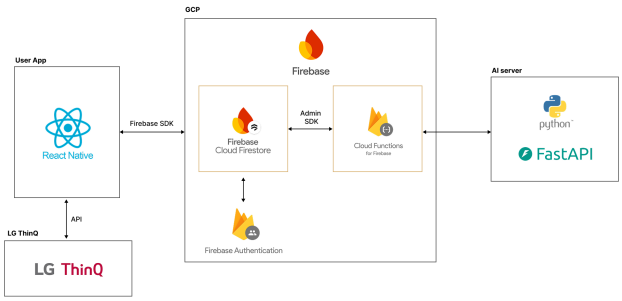


Fig. 17. Design Architecture

- Design tool: Figma
- Summary of features:
  - Provides the mobile interface for users to log in, sign up, and access the main HomeQuest experience.
  - Displays personalized challenges recommended by the AI server and allows users to accept missions.
  - Offers real-time progress tracking for both personal and family challenges.
  - Shows accumulated points and enables users to redeem rewards such as coupons.
  - Synchronizes continuously with Firebase to update family rankings, scores, and challenge status.

2) Backend
- Infrastructure: Firebase (Firestore, Authentication, Cloud Functions)
- Database: Firestore (NoSQL)
- Authentication: Firebase Authentication (Email/-Password)
- Cloud Functions: Event triggers, scheduled tasks, and API endpoints
- Summary of features:
  - Handles user registration and login through Firebase Authentication.
  - Manages all challenge-related operations: creating personal/family challenge progress documents, updating challenge status, calculating and storing personal/family points.
  - Manages family data: creating family documents and linking users to families, updating total family points in real time.
  - Listens to Firestore document changes (challenge completion, IoT-triggered events).
  - Provides backend business logic via Cloud Functions (ranking, point aggregation, event handling).

3) AI
- Framework: Python (scikit-learn)
- Web Framework: FastAPI
- AI Model: Gradient Boosting Decision Tree (GBDT)
- Summary of features:
  - Learns user and family behavior patterns from activity logs and ranks challenges by predicted success likelihood.

B. Overall Directory Organization

The HomeQuest project consists of four GitHub repositories: Frontend, Backend, AI, and Document. Each repository plays a distinct role in the project's architecture and development.

The frontend repository is responsible for implementing the mobile user interface and overall application flow of the service using React Native and Expo. It delivers a unified experience across Android and iOS, enabling users to authenticate, explore personalized challenges, accept missions, monitor progress, and view family rankings in real time. The src directory contains the core logic for screen navigation, state management, UI components, and gesture-based interactions, while continuous communication with Firebase and the AI recommendation server ensures that challenge lists, points, and family scores remain up to date.

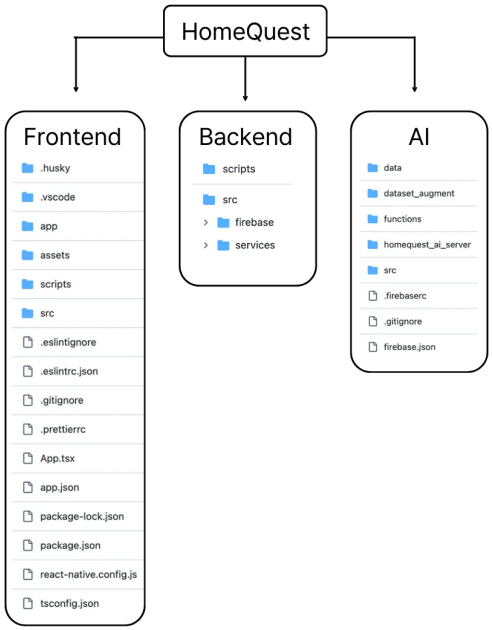


Fig. 18. Overall Directory

The backend repository manages all core data operations of the service, handling secure user authentication, family membership management, and the lifecycle of every challenge through Firebase Authentication and Cloud Firestore. It creates and maintains user documents, processes challenge acceptance and completion, updates personal and family points, and ensures that progress is consistently synchronized across clients. With Firestore's real-time listeners, all changes—such as mission updates and ranking shifts—are immediately reflected on the frontend, enabling a seamless and continuously up-to-date user experience.

The AI repository is dedicated to developing the system's core recommendation intelligence. It manages the full pipeline required to transform raw behavioral data into personalized challenge suggestions for each family. This repository maintains datasets, preprocessing logic, model development scripts, and the production-ready server that delivers real-time recommendations. The data directories support the preparation and simulation of long-term activity logs, which are used to train and validate the machine-learning models. The model development modules contain iterative versions of the recommendation algorithms, capturing the progression of feature engineering, scoring strategies, and performance tuning.

C. Module 1: Frontend

TABLE. Frontend Directory

Directory	File Name
HOMEQUEST_FrontEnd/	.expo/ .husky/ .vscode/ android/ ios/ app/ assets/ node_modules/ src/ .eslintignore .eslintrc.json .gitignore .prettierrc app.json App.tsx expo-env.d.ts package.json package-lock.json react-native.config.js tsconfig.json
HOMEQUEST_FrontEnd/app/	_layout.tsx +html.tsx +not-found.tsx Character.tsx index.tsx login.tsx modal.tsx onboarding-avatar.tsx onboarding-family.tsx onboarding-profile.tsx Setting.tsx signup.tsx
HOMEQUEST_FrontEnd/app/(tabs)	_layout.tsx four.tsx index.tsx market.tsx reward.tsx two.tsx
/src/components	tests__ EditScreenInfo.tsx ExternalLink.tsx StyledText.tsx Themed.tsx useClientOnlyValue.ts useClientOnlyValue.web.ts useColorScheme.ts useColorScheme.web.ts
/src/constants	colors.ts
/src/firebase	firebase.ts
/src/navigation	RootNavigator.tsx

1) Purpose
The HomeQuest frontend module, built with React Native and Expo, provides the primary user interface through which family members interact with all features of the app, including challenges, rewards, and rankings. It handles key screen transitions and gesture-based interactions such as browsing recommended challenges, accepting missions, and checking progress. The frontend also communicates with the Firebase backend and the AI recommendation server, receiving personalized challenge lists, points, and family rankings in real time based on user, family, and device activity logs, and rendering them seamlessly on the screen.

Directory	File Name
/src/screens	AuthDebugScreen.tsx Challenge.tsx CharacterScreen_2.tsx CharacterScreen.tsx HomeScreen.tsx MarketScreen.tsx OnboardingAvatar.tsx OnboardingFamily.tsx OnboardingProfile.tsx ranking.tsx RewardScreen.tsx SettingScreen.tsx Splash.tsx
HOMEQUEST_FrontEnd/src/store	index.ts useAuthStore.ts useChallengeStore.tsx useRewardStore.ts
HOMEQUEST_FrontEnd/src/types	challenge.ts index.ts
HOMEQUEST_FrontEnd/src/utils	index.ts

2) Functionality
The HomeQuest app's frontend allows users to log in, sign up, and seamlessly access the main experience of the service. Once authenticated, the app presents an interactive interface where users can explore personalized challenges, accept missions tailored to their family's lifestyle, and track their progress in real time. Users can also view their accumulated points and redeem them for coupons through an intuitive reward interface. The frontend continuously synchronizes with Firebase to retrieve updated family scores and rankings, ensuring that users always see the latest information.

3) Location of source code
https://github.com/CSE-HOMEQUEST/HOMEQUEST_FrontEnd

4) Class Components
• HOMEQUEST_FrontEnd/ : This folder contains the entire frontend source code of the HomeQuest application.
- HOMEQUEST_FrontEnd/ios/: iOS-specific native code and Xcode project files.
- HOMEQUEST_FrontEnd/assets/: Static resources such as images, icons, illustrations, and fonts used throughout the application's UI.
- HOMEQUEST_FrontEnd/app.json: The main Expo configuration file. Defines app metadata such as the app name, icon, splash screen, versioning, platform settings, and permissions.
- HOMEQUEST_FrontEnd/App.tsx: The root entry point of the React Native application. Initializes the app environment and loads the navigation or routing container.
- HOMEQUEST_FrontEnd/package.json: Contains metadata about the project along with lists of dependencies, development dependencies, and scripts (e.g., expo start, npm run build).
- tsconfig.json: TypeScript configuration file specifying compiler options, path aliases (e.g., @/src), and type checking rules.

• HOMEQUEST_FrontEnd/app/: This folder contains the top-level route definitions and screens managed by Expo Router. It defines the overall navigation structure of the HomeQuest app and maps URL-like paths to React Native screens.
- _layout.tsx: Defines the root layout for the app's main stack.
- +html.tsx: Special Expo Router route used for HTML rendering support or web-specific behavior when the app is built for web targets.
- +not-found.tsx: Default fallback screen rendered when a user navigates to an undefined route. It handles 404-style navigation errors within the Expo Router structure.
- Character.tsx: Screen that displays the detailed view of a selected family character.
- index.tsx: Top-level index route for the root app path.
- login.tsx: Screen where users can enter their credentials to log in.
- signup.tsx: Screen for new user registration.
- modal.tsx: A template modal screen generated by the Expo Router starter project.
- onboarding-avatar.tsx: Onboarding screen that lets users select their avatar when they first join HomeQuest.
- onboarding-family.tsx: Onboarding screen that lets users join an existing family by entering a family code or create a new family by generating their own code.
- onboarding-profile.tsx: Onboarding screen for setting up nickname and position within the family.
- Setting.tsx: Screen that provides application and account settings.

• HOMEQUEST_FrontEnd/app/(tabs)/: Contains the tab-based navigation group for the core in-app experience.
- layout.tsx: Configures the tab navigator layout, including which routes appear as tabs and how the tab bar is rendered.
- index.tsx: Tab screen for the the main Home section.
- two.tsx: Tab screen for the challenge section.
- reward.tsx: Tab screen for the Reward section.
- market.tsx: Tab screen for the Market section.
- four.tsx: Tab screen for the family ranking tab screen.

• HOMEQUEST_FrontEnd/src/: contains the core logic, UI components, navigation structure, state management, and utility modules for the HomeQuest application.

• HOMEQUEST_FrontEnd/src/components: This folder contains reusable UI components, shared hooks, and supporting utilities used throughout the app.

- tests/StyledText-test.js: A test file validating the behavior and rendering of the StyledText component.
- EditScreenInfo.tsx: A demo-oriented component from the Expo template that displays file path information and additional development details.
- ExternalLink.tsx: A component rendering external hyperlinks inside the application.
- StyledText.tsx: A custom text component applying consistent typography and styling across screens.
- Themed.tsx: Provides themed versions of View and Text, automatically adapting to light or dark mode based on the user's device settings.
- useClientOnlyValue.ts, useClientOnlyValue.web.ts: Hooks that return different values depending on whether the code is running on native or web.
- useColorScheme.ts, useColorScheme.web.ts: Hooks for retrieving the device's color scheme to dynamically apply theme logic.

• HOMEQUEST_FrontEnd/src/constants: Stores application-wide constant values.
- colors.ts: Defines the primary color palette used throughout HomeQuest for consistent UI styling.

• HOMEQUEST_FrontEnd/src/navigation: Contains navigation-related components and routing configuration.
- RootNavigator.tsx: Defines the main stack navigator for routing between screens such as onboarding, splash, and other non-tab sections.

• HOMEQUEST_FrontEnd/src/screens: This folder includes all main screens that make up the HomeQuest user interface.
- HomeScreen.tsx: Displays the home dashboard.
- Challenge.tsx: Shows available challenges or detailed challenge information for the user.
- RewardScreen.tsx: Displays the user's accumulated personal points and the family's total points.
- MarketScreen.tsx: Displays vailable coupons and redeem them using their accumulated points.
- Ranking.tsx: Presents the family leaderboard, showcasing rankings and accumulated family scores.
- CharacterScreen.tsx: Character detail screens that display avatar visuals, profiles and activity stats.
- OnboardingAvatar.tsx: Onboarding screen for selecting the user's avatar.
- OnboardingFamily.tsx: Onboarding screen for joining an existing family using a family code or creating a new family group.
- OnboardingProfile.tsx: Onboarding screen for setting the user's nickname and defining their position within the family.
- SettingScreen.tsx: Provides user account settings, app preferences, and profile management.
- Splash.tsx: Initial loading screen displayed during app startup.

• HOMEQUEST_FrontEnd/src/store: Contains global state management using Zustand.
- useAuthStore.ts: Maintains authentication state, user session data, and token information.
- useChallengeStore.tsx: Manages states related to challenges, including accepted and ongoing missions.
- useRewardStore.ts: Stores reward-related data, including earned points, redeemed rewards, and coupon history.

• HOMEQUEST_FrontEnd/src/types: Defines TypeScript interfaces and types used across the application.
- challenge.ts: Declares types and interfaces for challenge data structures.

D. Module 2: Backend

TABLE. Backend Directory

Directory	File Name
HOMEQUEST_FrontEnd/src/firebase/	firebase.ts
HOMEQUEST_FrontEnd/src/services/	authService.ts challengeService.ts rankingService.ts rewardService.ts marketService.ts
HOMEQUEST_FrontEnd/scripts/dataset/	families.xlsx users.xlsx challenges.xlsx challenge_events.xlsx seedFirestore.js challenge_progress_seedFirestore2.js

1) Purpose
The backend module securely manages all user and family data through Firebase Authentication and Cloud Firestore. It processes user sign-up and login, creates the corresponding `/users/{userId}` document, and maintains family membership information.
The module handles end-to-end challenge functionality, including accepting a challenge, creating a `challengeProgress` document, updating completion status, and automatically accumulating personal and family points. Real-time updates are triggered through Firestore listeners, allowing the frontend to reflect progress and rankings without manual refresh.

2) Functionality
The backend module provides the following key functionality:

- **Authentication and user profile management**: Using Firebase Authentication, `authService` handles sign-up, login, and logout. When a new user signs up, it also creates a corresponding `/users/{userId}` document in Firestore to store profile data such as nickname, family ID, and role within the family.

- **Challenge management**: `challengeService` reads challenge templates from the `/challenges` collection, applies filters such as mode (personal/-family) and category (chores, health, saving), and returns typed `ChallengeDto` objects. It supports pagination using `limit` and `startAfter`, and will be extended to record accept and complete events.
- **Ranking and family scores**: `rankingService` will aggregate points stored in `/families` and `/users` to compute both family rankings and personal rankings. It will provide functions to fetch the current family leaderboard and to update scores when a challenge is completed.
- **Reward and market logic**: `rewardService` and `marketService` will handle viewing available rewards, redeeming coupons, and consuming items from the `/products` and `/randomBoxes` collections. They will ensure that user points are updated atomically and that redemption history is stored for transparency.
- **Data seeding and admin scripts**: For testing and demonstration, the `dataset` folder contains seed scripts (using the Firebase Admin SDK or Python) that upload initial data for families, users, challenges, challenge events into Firestore.

Overall, the backend centralizes all Firestore and Firebase Auth access in the service layer so that the frontend only needs to call intuitive APIs such as `apiLogin`, `fetchChallenges`, `getRanking`, or `redeemReward`.

3) Location of source code
   Repository: https://github.com/CSE-HOMEQUEST/HOMEQUEST_FrontEnd

4) Class Components
   • **HOMEQUEST_FrontEnd/src/firebase/:** This folder encapsulates Firebase client initialization and exports reusable instances for the rest of the app.
   - `firebase.ts`: Initializes Firebase using the project configuration and exports the configured `auth` (Firebase Authentication) and `db` (Cloud Firestore) instances. All backend service modules import these instances rather than creating their own.

   • **HOMEQUEST_FrontEnd/src/services/:** This folder contains the backend service layer. Each file provides a cohesive set of functions that hide raw Firestore queries and authentication APIs, exposing a clean, typed interface to the frontend.
   - `authService.ts`: Manages authentication processes such as sign-up, login, and logout using Firebase Authentication. It also creates and updates the corresponding `/users/{userId}` document in Firestore so that profile and family data are kept in sync with the authentication state.
   - `challengeService.ts`: Manages the full lifecycle of challenges within the Firebase backend. When a user accepts a challenge, the service retrieves the corresponding template from the collection `/challenges` and refines the metadata so it can be stored as a structured progress document. For personal challenges, a new progress record is created under `/users/{userId}/challengeProgress/{progressId}`. For family challenges, the service additionally creates a parallel document at `/families/{familyId}/challengeProgress/{progressId}` so that individual and family-level progress remain synchronized.
     As the user advances through the challenge, the progress document is updated to reflect its current status. When a challenge is completed, the service updates the status field and calculates the earned points, applying them atomically to both `/users/{userId}` (personal points) and `/families/{familyId}` (family total points). These document updates automatically trigger Firestore real-time listeners, ensuring that family rankings, personal scores, and challenge progress are immediately reflected across the application. Through this workflow, the `challengeService` abstracts all low-level Firestore operations and provides a clean, consistent interface for retrieving challenge data, creating progress documents, tracking states, and synchronizing user–family point updates.
   - `rankingService.ts` : Provides ranking and leaderboard functionality. It will aggregate points from `/families` and `/users` to compute family rankings and personal rankings, and expose functions such as `getFamilyRanking()` and `getPersonalRanking(familyId)`.
   - `rewardService.ts` : Manages reward data and redemption logic. It will provide functions like `getAvailableRewards()` to load rewards, `redeemReward(userId, rewardId)` to deduct points and record redemption history, and `getUserRedemptions(userId)` to retrieve past redemptions.
   - `marketService.ts` : Provides data and operations for the in-app market and random boxes. It will load items from `/products` and `/randomBoxes`, implement `openRandomBox(userId, boxId)` to grant random rewards, and return detailed product information.

• **HOMEQUEST_FrontEnd/scripts/:** This folder contains scripts and data files used to seed Firestore with initial data for testing, feature development, and demonstration.

- `families.xlsx`: Contains sample family documents, including family names, generated family codes, and initial total family points. Used to populate the `/families` collection for testing family creation and onboarding flows.
- `users.xlsx`: Provides sample user profiles such as nicknames, avatar selections, family roles, and associated family IDs. Used to seed the `/users` collection for authentication and user–family linking tests.
- `challenges.xlsx`: Defines challenge templates used by the app, including title, category (chores/health/saving), mode (personal/family), base points, and metadata. Populates the `/challenges` collection.
- `challenge_events.xlsx`: Contains sample challenge accept and completion events, storing timestamps, user IDs, family IDs, and earned points. Used to seed the `/challengeEvents` subcollections for ranking, progress tracking, and analytics.
- `seedFirestore.js`: A Node-based Firebase Admin SDK script that reads the Excel files above and uploads their contents to Firestore. This script automates initial dataset generation for development and testing.
- `challenge_events_seedFirestore2.js`: A Node-based script built on the Firebase Admin SDK that reads the Excel files described above and uploads their contents to Cloud Firestore. During this process, it transforms the structured data columns into realistic log documents under `/users/{userId}/challengeProgress` and `/families/{familyId}/challengeProgress`, so that each row in the dataset becomes a properly formatted contribution record reflecting challenge participation, progress updates, and earned points for both individual users and families.

E. Module 3: AI

TABLE. HOMEQUEST_AI Directory

Directory	File Name
HOMEQUEST_AI/data/raw/	challenges.xlsx challenge_events.csv challenge_events.xlsx challenge_events_seed.py families.xlsx users.xlsx
HOMEQUEST_AI/data/processed/	homequest_simulated_6months.csv
HOMEQUEST_AI/dataset_augment/	augment_dataset.py homequest_simulated_10k.csv
HOMEQUEST_AI/functions/	.gitignore index.js package.json package-lock.json
HOMEQUEST_AI/homequest_ai_server/	homequest_ai_final.py main.py recommend_counts.json
HOMEQUEST_AI/src/models/	AI code-1.py AI code-2.py AI code-3.py AI code-4.py AI code-5.py AI code-6.py AI code-7.py
HOMEQUEST_AI/src/state/	recommend_counts.json

1) Purpose

The AI Recommendation Module serves as a core component of the HomeQuest service. Its primary objective is to analyze various activity records generated within the household and automatically propose suitable challenges for the user. Rather than relying on fixed rules or random selection, the module evaluates daily behavior patterns and past habits to identify which challenge the user is most likely to complete on a given day.

To achieve this, the module extracts multiple features from the activity data and uses them as inputs to a machine learning model that predicts the likelihood of completing each challenge. The resulting scores reflect both user-specific conditions and challenge attributes, ensuring that the recommendations align naturally with the user's daily routine.

In addition, the module incorporates mechanisms such as recommendation-frequency adjustment and cooldown checks to prevent excessive repetition of the same challenge. These controls enhance diversity and maintain the user's engagement with different types of challenges, contributing to a balanced and dynamic recommendation flow.

Ultimately, this module enables HomeQuest to function not only as an automation tool but also as a system that encourages consistent participation and behavioral improvement. It plays a central role in guiding users toward sustainable habits by offering personalized and context-aware daily goals.

2) Functionality

The AI Recommendation Module operates through the following key functionalities.

First, it loads activity data and challenge metadata, then restructures them into formats suitable for model processing. The module extracts variables such as weekday,

personal points, family points, energy usage, mode, and category. Categorical values undergo one-hot encoding, and for speed-type challenges, notification and completion timestamps are converted into minutes to calculate the time difference and determine whether the task was completed within 60 minutes.
Second, the module trains two GradientBoostingClassifier models. One model predicts the completion of general challenges, while the other predicts whether a speed-type challenge can be finished within one hour. The dataset is divided into training and evaluation segments based on a predefined range, and the models' performance is assessed through AUC values.
Third, the trained models are used to evaluate which challenges are suitable for recommendation on the current day. Completion probabilities are calculated using user and challenge attributes, and these scores are adjusted using past recommendation frequency. Cooldown rules defined for each challenge are also checked to determine whether the challenge can be recommended today.
Fourth, the module generates recommendations separately for daily, monthly, and speed challenges. It prioritizes candidates with higher scores and uses a probability-based selection method to prevent the same challenge from appearing repeatedly. For speed challenges, multiple notification times are simulated, and the most favorable time slot is selected along with the challenge itself.
Finally, the selected challenge is recorded in a JSON file, and this history is incorporated into future score adjustments. This process reduces repetitive recommendations and helps maintain balanced variety across different challenge types.

3) Location of source code
https://github.com/CSE-HOMEQUEST/HOMEQUEST_AI

4) Class Components
- **HOMEQUEST_AI/data/**: This folder contains all datasets used for modeling, simulation, and backend integration in the HomeQuest AI module. It is divided into `raw/` (source data) and `processed/` (cleaned and transformed data).
  - **HOMEQUEST_AI/data/raw/**: Stores original, unmodified source datasets used as inputs for preprocessing, analysis, and AI model training.
    * `challenges.xlsx`: Master list of HomeQuest challenges, including challenge IDs, categories, modes, and metadata required for recommendation logic.
    * `challenge_events.csv`: Raw event log containing user challenge executions, timestamps, and completion status. Used for exploratory analysis and baseline statistics.
    * `challenge_events.xlsx`: Spreadsheet version of the event logs for manual inspection, debugging, and annotation.
    * `challenge_events_seed.py`: Python script for generating synthetic challenge event data, enabling recommendation algorithm testing without real IoT input.
    * `families.xlsx`: Dataset defining simulated family groups, family IDs, and structural attributes for group-based behavior analysis.
    * `users.xlsx`: User profile dataset linking individuals to events, challenges, and behavioral metrics.
  - **HOMEQUEST_AI/data/processed/**: Contains cleaned, merged, and feature-engineered datasets derived from raw data.
    * `homequest_simulated_6months.csv`: Six-month synthetic dataset combining user, family, and challenge logs, used for model training, behavior simulation, and backend testing.
  - **HOMEQUEST_AI/dataset_augment/**: Contains files generated for synthetic dataset creation using the OpenAI API to improve model realism and behavioral diversity.
    * `augment_dataset.py`: Script that uses the OpenAI GPT-4.0 Mini API to generate human-like smart-home behavior patterns and convert them into structured event logs.
    * `homequest_simulated_10k.csv`: Augmented dataset (10,000 lines) produced via the script and used for training and validating the recommendation model, improving generalization and performance in real-world conditions.
- **HOMEQUEST_AI/functions/**: Contains core server-side scripts and configuration files used to run Cloud Functions and backend logic for the HomeQuest system.
  - `.gitignore`: Specifies which files and directories should be excluded from version control, preventing unnecessary or sensitive resources from being tracked.
  - `index.js`: Primary entry point for the Cloud Functions service. Defines exported functions, HTTP endpoints, and backend logic that interacts with Firebase, Firestore, or external APIs.

  - `package.json`: Contains metadata, scripts, dependencies, and engine settings required to configure and run the backend environment.
  - `package-lock.json`: Stores the exact dependency tree of all installed packages, ensuring reproducible and stable deployments.

- **HOMEQUEST_AI/homequest_ai_server/**: This folder contains the AI logic, model pipeline, and execution scripts for the HomeQuest recommendation engine.
  - `homequest_ai_final.py`: Main AI engine that implements data preprocessing, feature extraction, scoring logic, and the final challenge recommendation algorithm.
  - `main.py`: Execution entry point for the AI server. Loads the model and datasets, defines API routes (e.g., FastAPI endpoints), receives external requests, and returns recommended challenges.
  - `recommend_counts.json`: Lightweight persistence file that tracks how often each challenge has been recommended, supporting cooldown rules and diversity balancing.

- **HOMEQUEST_AI/src/**: Contains the core AI model codebase, experimental model versions, and state management modules for the recommendation system.
  - **HOMEQUEST_AI/src/models/**: Holds multiple development iterations and experimental versions of the recommendation algorithm.
    * `AI code-1.py`: Early prototype featuring basic preprocessing and simple rule-based scoring.
    * `AI code-2.py`: Speed-optimized version focusing on reducing inference latency through lightweight transformations.
    * `AI code-3.py`: Refined near-final implementation with improved feature engineering and scoring logic.
    * `AI code-4.py`: Variant exploring enhanced weekly/daily behavior modeling and alternative evaluation metrics.
    * `AI code-5.py`: Version tuned for better probability calibration and balanced challenge category distribution.
    * `AI code-6.py`: Stable reproducible model trained on a fixed dataset for consistent benchmarking.
    * `AI code-7.py`: Latest consolidated model iteration integrating improvements from previous versions for production alignment.
  - **HOMEQUEST_AI/src/state/**: Contains persistent files used to regulate AI behavior over time.
    * `recommend_counts.json`: Tracks challenge recommendation frequency to prevent repetition, enforce cooldowns, and maintain diversity in suggested challenges.

VI. USE CASE

A. Loading

Fig. 19. Loading

When the application is launched, the HomeQuest loading screen briefly appears. This screen is displayed only while the system prepares the necessary components for operation. Once initialization is complete, it automatically transitions to the next page.

B. Sign Up

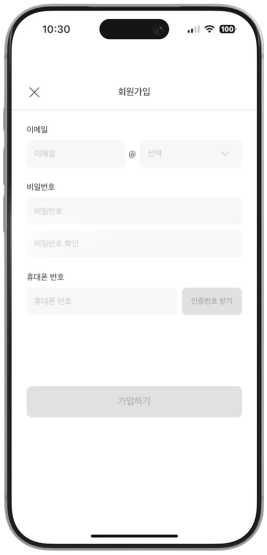


Fig. 20. Sign Up

Users can register by entering their basic information, including an ID, password, email address, and phone number. After completing the registration process, they are redirected to the Login page, where they can sign in with their newly created account.

C. Login

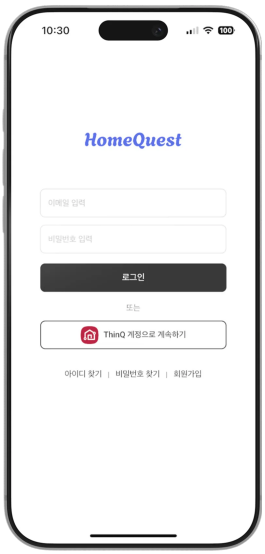


Fig. 21. Login

On the Login page, users can access their accounts by entering their registered ID and password. A login option using a ThinQ account is also provided.

D. Onboarding

Fig. 22. basic Information Setup

Onboarding occurs only at first login and includes the following steps:

1) Basic Information Setup : Users enter their nickname and family role, then grant location permission to set their family's home location.

Fig. 23. Avatar Creation

2) Avatar Creation : Users select an avatar design from available options to visually represent themselves.

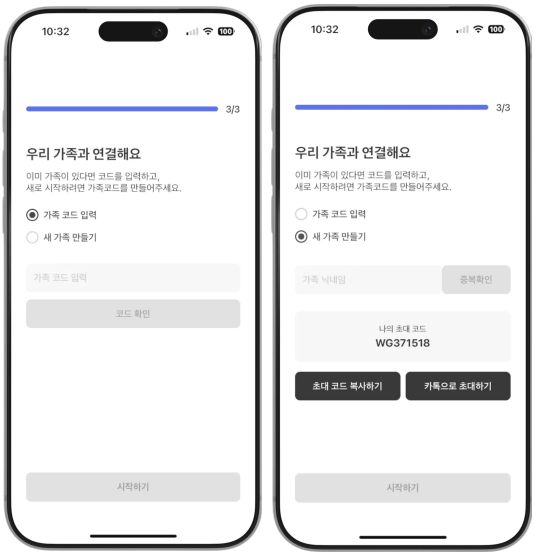


Fig. 24. Family Connection

3) Family Connection : Users may enter an invitation code to join an existing family group. If creating a new group, users set a family nickname and share the invitation code to invite others.

E. Main Page

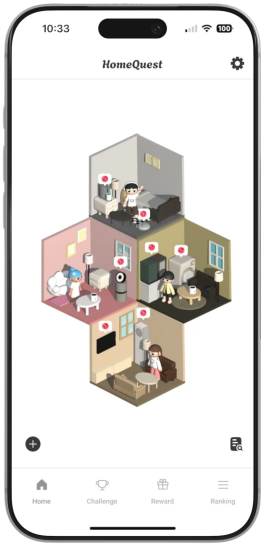


Fig. 25. Home

After completing onboarding or logging in, users are directed to the Main Page. This page displays the home structure containing all invited family members.

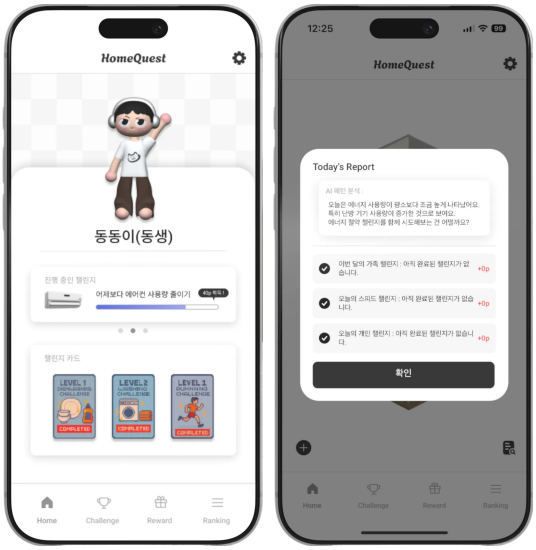


Fig. 26. Character Detail & Report

- Users can tap a character icon to view detailed profile information.
- The plus (+) button allows users to add new rooms or register furniture and home appliances.
- The report button provides access to challenge completion history and AI-generated summary insights.

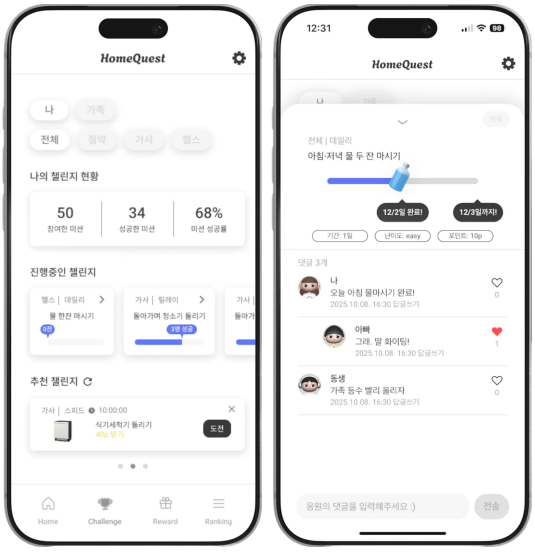


Fig. 27. Challenge

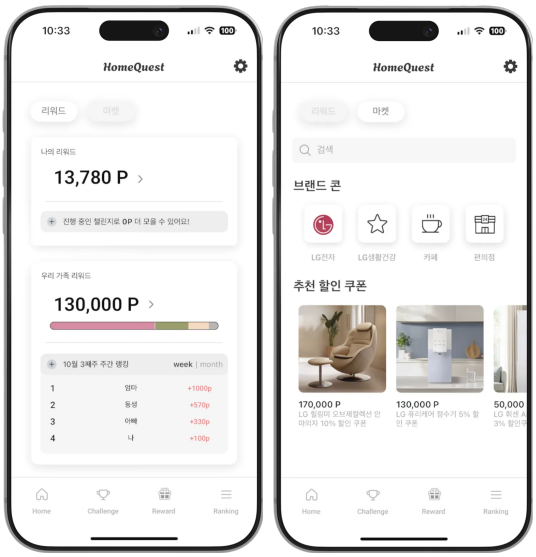


Fig. 28. Reward

F. Challenge Page

Users can access the Challenge Page to view their overall challenge progress.

- Selecting an ongoing challenge displays details such as activity dates, progress rate, challenge duration, difficulty mode, and obtainable points.
- A comment and like feature allows users to support and interact with one another.
- When a recommended challenge is presented, users may accept and begin immediately, decline, or request another suggestion using the “Recommend Again” option.

G. Reward Page

Users can view and use points through the Reward tab and Market tab.

1) Reward Tab
   - Users can check their personal points earned from individual challenges and family points obtained from family challenges.
   - Detailed pages display earning and usage history in a timeline format.
   - Family points include contribution indicators and weekly/monthly ranking progress for each member.

2) Market Tab
   - Users can redeem LG-related coupons using family points, while personal points can be used to purchase everyday coupons such as café or convenience store vouchers.
   - Search functions, brand filters, and recommended coupon sections are also available.

H. Ranking Page

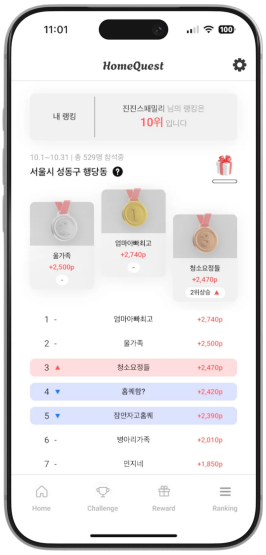


Fig. 29. Ranking

Users can check family rankings based on their local area for a selected period.

1) Reward Box
   - As the family’s ranking increases, the reward box gauge fills.
   - Once fully charged, users can receive random rewards such as discount coupons or bonus points.

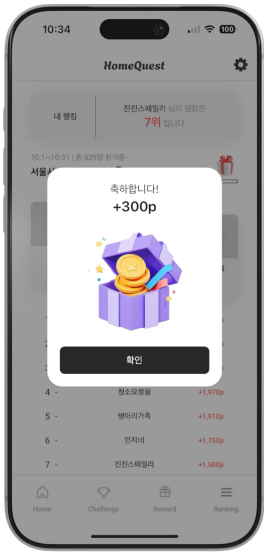


Fig. 30. Reward Box

VII. Discussion

The proposed HomeQuest system demonstrates the potential of integrating IoT data and behavioral gamification to promote family engagement and positive household habits. However, several practical and architectural limitations remain, which highlight key challenges for real-world deployment.

First, the feasibility of full-scale IoT integration is constrained by the accuracy and granularity of ThinQ-based device data. Because all behavioral signals depend on device-level event logs, variability in sensing frequency and data quality directly affects model reliability. Furthermore, security and data-fusion constraints limit how closely device logs can be linked with user-level behavior. As a result, complete end-to-end coupling between real IoT streams and the behavior model may not be achievable under the current architecture.

Second, sustaining long-term behavior change remains a fundamental challenge. As with many gamified systems, initial novelty and enjoyment tend to diminish over time, potentially causing engagement fatigue. The system also relies on consistent participation from multiple family members to maintain its feedback loop. If engagement decreases for even a single member, the motivational effect of the collective challenge weakens significantly.

Third, family heterogeneity introduces limitations for personalization. Families differ widely in lifestyle patterns, device ownership, daily routines, and task distribution. While the current recommendation model provides a baseline level of personalization, machine learning approaches—particularly lightweight models such as the LightGBM baseline used here—have inherent limitations. These models may not fully capture the complex and diverse behaviors exhibited across different households, restricting the accuracy of challenge recommendations.

Finally, the scope of IoT-detectable behaviors is fundamentally limited. Many meaningful household actions cannot be captured through IoT devices alone, such as cleaning tasks, organizing activities, or shared family moments. These behaviors require hybrid verification methods, including photo evidence, push confirmations, or proximity sensing. Consequently, IoT-only tracking is insufficient for representing the full spectrum of household behaviors, and a mixed-modal authentication approach is necessary for robust and equitable challenge evaluation.

References

[1] Samsung SmartThings Energy. Available: https://www.smartthings.com/energy
[2] Strive App. Available: https://www.striveapp.com
[3] FamilyAlbum by Mixi. Available: https://family-album.com/
[4] Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. Proceedings of the 19th International Conference on World Wide Web (WWW 2010), 661–670. Available: https://doi.org/10.1145/1772690.1772758