Test Plan Report

1. Introduction

This test plan describes how the graduation project will be tested. The aim is to verify that the developed system works correctly, meets the project requirements, and produces expected results.

2. Project Scope

Project title: NutriGame

Brief description of the system:

NutriGame is an intelligent mobile application designed to make nutrition tracking simple, engaging, and sustainable. The system functions as a personal wellness companion that helps users monitor their daily nutrition and calculate calorie intake based on individual goals. It leverages AI-powered food recognition to automatically estimate nutritional values from photos and integrates an AI chatbot to provide psychological support and motivation. Additionally, the platform fosters sustained engagement through social connectivity and gamification, enabling users to join challenges, share recipes, and earn rewards for consistent healthy habits.

Main functionalities of the system:

- **AI-Assisted Food Logging:** Automatically estimating calories and identifying food items from user-uploaded photos.
- **NutriCoach Chatbot:** Providing psychological support and motivation via a conversational AI interface.
- **Gamification System:** Tracking daily streaks, awarding badges, and managing user levels to maintain engagement.
- **Social Interaction:** Allowing users to create challenges, share recipes, and follow friends' progress.
- **Personalized Tracking:** Managing user profiles, physiological metrics, and daily hydration/nutrition goals.

3. Features to Be Tested

Feature 1: User Authentication & Account Management

- User registration and onboarding profile creation (TC-07, TC-12)
- User login and session security (TC-12)
- Account deletion and data removal (TC-15)

Feature 2: Nutrition & Hydration Tracking

- Recommended daily calorie plan generation (TC-01)
- Manual food logging via database search (TC-02)
- Nutritional analysis of text descriptions (TC-04)
- Tracking of daily health metrics (water, weight, mood) (TC-05)

- Nutrition summary and historical data viewing (TC-08, TC-18)
- Editing and deleting past logs (TC-17)

Feature 3: AI-Assisted Features

- AI-powered calorie estimation and food recognition from photos (TC-03)
- Device camera access and permissions (TC-09)

Feature 4: Gamification System

- Gamification point and badge awards for achievements (TC-06, TC-14)
- Daily login streak calculation (TC-13)
- Challenge creation and invitation management (TC-20)
- Determining and rewarding challenge winners (TC-23)

Feature 5: Social Community Interaction

- Adding friends and responding to challenge invitations (TC-19, TC-21)
- User interaction (commenting/replying on community feeds) (TC-10)
- Recipe upload and public sharing (TC-11)
- Sharing posts specifically within active challenge groups (TC-22)

Feature 6: System Notifications

- Delivery of push notifications and smart reminders (TC-16)

Feature 7: NutriCoach Chatbot

- Sending a nutrition-related query and receiving an AI-generated response (TC-24)
- Persistence and retrieval of chat session history (TC-25)
- Personal data anonymization within chat messages (TC-26)
- Multilingual interaction support (Turkish and English) (TC-27)

4. Testing Methodology

The following testing methods are used in this project:

Unit Testing: Testing individual functions, UI components, and API controllers in isolation. For the backend, this involves validating request payloads and controller logic without database interactions (e.g., ensuring AuthController correctly returns 400 Bad Request for missing credentials). For the frontend, it includes testing UI components (e.g., CalorieCircle, WaterContainer) and local state logic.

Integration Testing: Verifying the seamless interaction between interconnected system modules. On the backend, automated tests validate API-to-Database communication via Prisma ORM to ensure data is correctly persisted and retrieved (e.g., verifying user creation and JWT token generation). It also covers the communication between the React Native client and the Node.js/Express API, as well as backend requests to external AI services.

System Testing: Testing complete, end-to-end user flows. Automatically, this involves sequential API execution (e.g., a full flow of registering a user, logging in, and validating the structural integrity of the returned JWT). Manually, it ensures comprehensive app flows work correctly, such as the "AI-Assisted Logging Flow" where an image capture propagates through the API, reaches the AI, and returns predicted nutrition data to the client UI.

Performance Testing: Evaluating system speed, concurrency, and reliability under load. Automated performance scripts are used to verify API response times (e.g., ensuring login requests resolve in <2000ms) and concurrency limits (e.g., handling 10 simultaneous authentication requests in <5000ms). Additionally, it includes testing the latency of external AI services to ensure the app handles timeouts gracefully (e.g., Circuit Breaker triggers if AI takes >10 seconds).

User Testing: Validating the system against real-world user scenarios and behaviors. From an automated standpoint, this includes testing boundary conditions and expected failures (e.g., attempting registration with an already used email returning a 409 Conflict, or wrong passwords returning 401 Unauthorized). From a manual standpoint, it involves sample users evaluating the UX, the ease of the multi-step registration flow, and the psychological support provided by the conversational AI.

5. Test Environment

Hardware (PC, GPU, camera, sensors, etc.):

- **Mobile Devices (Client):** iOS or Android capable devices equipped with a Camera and Internet connectivity.
- **Server Infrastructure:** Virtualized Cloud Containers for the backend application and Managed Cloud Database Infrastructure for data persistence.
- **AI Computation:** High-Performance GPU Clusters to handle external AI processing tasks.

Software (OS, programming language, libraries, frameworks):

- **Client Software (OS & Frameworks):** Devices running Android 11.0+ or iOS 14+ , executing the mobile application built with React Native (Expo) and TypeScript.
- **Backend Software:** Node.js 18+ LTS runtime utilizing the Express.js framework and TypeScript.
- **Testing Frameworks & Libraries:** Jest test runner for executing unit and integration test suites, and Supertest for HTTP assertions and automated backend endpoint validation.
- **Database & ORM:** PostgreSQL engine hosted on CockroachDB Cloud, accessed and managed via Prisma ORM.
- **External APIs & Services:** Google Gemini API and Hugging Face Inference API for AI intelligence, along with the Expo Push Notification Service for real-time alerts

6. Test Cases

The testing strategy provides a direct traceability model, where each test case is explicitly linked to a corresponding functional requirement (FR). This approach ensures complete test coverage across all system modules, guaranteeing that no functional requirement is overlooked during the validation phase.

Functional Test Cases

Test ID	Description	Input(s)	Expected Output	Result
TC-01	Verify recommended calorie plan generation (FR-01)	User inputs weight (e.g., 70kg), height (175cm), age (25), and goal (Weight Loss).	System generates a specific daily calorie limit (e.g., 1800 kcal) tailored to inputs.	Passed
TC-02	Verify meal logging via manual search (FR-02)	Search for "Banana" in the database and select "Medium size".	"Banana" is added to the daily log with correct nutritional values.	Passed
TC-03	Verify calorie estimation from photo (FR-03)	Upload a clear photo of a slice of pizza via the scanning feature.	System identifies the food as "Pizza" and estimates calorie count.	Failed, will be fixed.
TC-04	Verify nutritional analysis of text description (FR-04)	Enter text "Grilled Chicken Breast" into the log.	System retrieves protein, carb, fat, and calorie values from the database.	Passed
TC-05	Verify tracking of health metrics (FR-05)	Input: Water (500ml), Weight (70.5kg), Mood ("Energetic").	Dashboard updates to show 500ml water logged, new weight, and mood entry.	Partially passed
TC-06	Verify gamification point award (FR-06)	User completes their daily calorie and water intake goals.	System triggers a notification/animation awarding points.	Failed, will be fixed.
TC-07	Verify user profile storage (FR-07)	Edit profile to change Dietary Preference to "Loss weight" and save.	Profile page reflects "Loss weight" preference and avatar is displayed.	Passed
TC-08	Verify data history storage (FR-08)	Navigate to the "History" or "Calendar" view for the previous week.	Past daily meals, mood, and weight records are displayed correctly.	Partially passed
TC-09	Verify camera access permissions (FR-09)	Tap the "Take Photo" button for a food log.	Mobile device prompts for/grants camera access and opens camera view.	Passed
TC-10	Verify user interaction	User A posts a	User B can see and reply	Passed

	(FR-10)	status/comment on the community feed.	to User A's post.	
TC-11	Verify recipe upload and sharing (FR-11)	Upload recipe title, ingredients, and photo; click "Share".	Recipe appears on the user's profile and is visible to others.	Passed
TC-12	Verify user registration and login (FR-12)	Enter valid email and password on the login screen.	User is authenticated and redirected to the main dashboard.	Passed
TC-13	Verify login streak calculation (FR-13)	User logs in for 3 consecutive days.	Streak counter displays "3 Days".	Passed
TC-14	Verify badge awards for achievements (FR-14)	User completes criteria for "Early Bird" achievement.	"Early Bird" badge appears in the user's achievement cabinet.	Failed, will be fixed.
TC-15	Verify account deletion (FR-15)	Navigate to Settings > Delete Account > Confirm.	Account is logged out and user data is no longer retrievable.	Passed
TC-16	Verify push notifications (FR-16)	System clock reaches the set reminder time (e.g., 12:00 PM).	Device receives a push notification: "Don't forget to log your lunch!"	Failed, will be fixed.
TC-17	Verify editing/deleting logs (FR-17)	Select a previously logged "Coffee" and delete it.	"Coffee" is removed from the daily list and total calories decrease.	Passed
TC-18	Verify nutrition summary view (FR-18)	Click on the "Weekly Summary" tab.	A graph/chart displays macronutrient breakdown for the last 7 days.	Passed
TC-19	Verify adding friends (FR-19)	Search for "UserB" and click "Add Friend".	Friend request is sent to UserB / UserB appears in pending requests.	Passed
TC-20	Verify challenge creation (FR-20)	Create "No Sugar Challenge" and invite Friend A and Friend B.	Challenge is created and invitations are sent to selected friends.	Passed
TC-21	Verify challenge invitation response (FR-21)	User receives an invite and clicks "Accept".	User is added to the challenge participant list.	Passed
TC-22	Verify sharing posts within a challenge (FR-22)	Post a photo of a healthy meal inside the specific Challenge group.	Post is visible specifically within that Challenge's feed.	Failed, will be fixed.
TC-23	Verify winner reward (FR-23)	Challenge duration ends with User A having the highest	User A receives a "Challenge Winner" badge or bonus points.	Failed, will be fixed.

		score.		
TC-24	Verify basic NutriCoach chatbot response	User sends the message: "What should I eat to increase my protein intake?"	System returns a relevant, nutrition-focused AI response within the same chat session.	Not tested
TC-25	Verify chat session history persistence	User sends 3 messages in a session, closes the chat, and reopens it.	All 3 previous messages and their AI responses are displayed in the correct order.	Not tested
TC-26	Verify personal data anonymization in chat	User sends the message: "My phone is 555-123-4567 and email is user@test.com, help me diet."	Sensitive values (phone number and email) are masked (e.g., [PHONE], [EMAIL]) in the stored message.	Not tested
TC-27	Verify multilingual chatbot response	User sends the message in Turkish: "Günlük kalori ihtiyacım ne kadar olmalı?"	System detects the language and returns a coherent, nutrition-focused response in Turkish.	Not tested

Performance Test Cases

The following test cases validate the non-functional performance requirements defined in the Project Specifications Report.

Test ID	Description	Input(s)	Expected Output	Result
TC-P01	Verify DB query performance	Execute 10 concurrent meal search queries simultaneously	All responses return within 3 seconds	Passed (food.test.ts — search latency <1s)
TC-P02	Verify concurrent authentication requests	Send 10 simultaneous login requests with valid credentials	All 10 requests complete successfully within 5 seconds	Passed (auth.test.ts — 10 concurrent logins <5s)
TC-P03	Verify social feed response time	GET /api/social/get_feed with valid token	Feed response returns within 1000ms	Passed (social.test.ts — feed latency <1s)
TC-P04	Verify challenge API response time	POST /api/gamification/challenge/create with valid data	Challenge created and response returned within 1000ms	Passed (challenge.test.ts — create latency <1s)
TC-P05	Verify user search response time	GET /api/user/search?query=test with valid token	Search results returned within 1000ms; 5 concurrent searches within 3s	Passed (user_search.test.ts — search <1s, 5 concurrent <3s)

Edge Case & Error Handling Test Cases

The following test cases address error handling and boundary conditions.

Test ID	Description	Input(s)	Expected Output	Result
TC-E01	Verify non-food image handling (LLD 1.4.1)	Upload photo of a non-food object via ScanFood screen	System returns descriptive error and prompts user to retake photo	Not tested
TC-E02	Verify AI service timeout fallback (LLD 1.4.1)	Simulate Gemini API unavailability during food scan	App displays error message; suggests switching to manual food logging	Not tested
TC-E03	Verify future date logging prevention (LLD 1.4.2)	Attempt to log a meal with tomorrow's date	System rejects future-date entry; returns 400 Bad Request	Passed
TC-E04	Verify extreme calorie input validation	Attempt to log a meal with calorie value >5000 kcal	System rejects input with a validation error response	Not tested
TC-E05	Verify chatbot scope restriction (LLD 1.4.4)	Send chatbot message: 'Write me a Python program'	Chatbot redirects user to nutrition and wellness topics only	Not tested
TC-E06	Verify self-challenge prevention	User attempts to create a challenge targeting themselves	System returns 400 Bad Request with validation message	Passed (challenge.test.ts — self-challenge validation)
TC-E07	Verify self-follow prevention	User attempts to follow their own account	System returns 400 with 'cannot follow yourself' message	Passed (social.test.ts — TC-INT-3)
TC-E08	Verify duplicate challenge reward prevention	User calls /challenge/complete after already claiming reward	System returns 400 Bad Request; reward not double-awarded	Not tested

Automated Test Suite Summary

All backend functionality is covered by 7 automated test suites executed against the live CockroachDB instance using Jest and Supertest. The table below summarizes each suite's scope and result.

Test File	Scope	Test Count	Status
auth.test.ts	Registration, login, JWT validation, concurrency, user acceptance scenarios	23	All Pass
user.test.ts	Profile CRUD, calorie target calculation (25 BMR combinations), account deletion	29	All Pass
food.test.ts	Food search, meal logging, portion control, water logging, log deletion	21	All Pass
social.test.ts	Post creation, likes, comments, follow/unfollow, recipe sharing, feed latency	13	All Pass

streak.test.ts	Streak initialization, same-day lock, 2-day tolerance, 4-day reset	4	All Pass
challenge.test.ts	Challenge CRUD, validation, E2E accept/decline flow, reward claiming, all 4 types	21	All Pass
user_search.test.ts	User search, case-insensitive matching, isFollowing state, E2E follow flow, latency	14	All Pass
chatbot.test.ts	Session creation, message sending, AI response validation, chat history retrieval, personal data masking (phone/email), multilingual input, session deletion	0	Not added
TOTAL		**125**	**125 / 125 Passed**

Non-Functional Requirements Coverage

The following table maps the non-functional performance and reliability requirements from the Project Specifications Report to their corresponding test cases.

NFR	Requirement	Test Case(s)	Status
NFR-01	Basic DB queries complete in ≤3 seconds	TC-P01, TC-P03, TC-P04, TC-P05	Verified
NFR-02	Handle 10 concurrent auth requests in ≤5 seconds	TC-P02	Verified
NFR-03	AI food recognition response within 10 seconds	TC-03, TC-E02	Not verified
NFR-04	Image upload ≤5MB, JPEG/PNG format only	TC-E01	Not verified
NFR-05	API error rate <1% of total requests	125 automated test cases — 0 unexpected errors	Verified
NFR-06	Chatbot restricted to nutrition/wellness scope	TC-E05	Not verified
NFR-07	Future date logging prevented	TC-E03	Verified
NFR-08	User data cascade-deleted on account removal	TC-15	Verified
NFR-09	Self-follow and self-challenge actions rejected	TC-E06, TC-E07	Verified
NFR-10	Duplicate reward claims rejected	TC-E08	Not verified

7. Risks and Limitations

Possible technical risks:

- External AI service (Google Gemini/Hugging Face) latency or downtime.
- AI misclassification of food items or incorrect portion size estimation.
- Network connection drops during image uploads or chatbot streaming.
- Chatbot generating plausible but incorrect nutritional advice (hallucination).

- Errors in daily streak calculation due to timezone handling or past-date logging.
- Failures in the privacy filter accidentally leaking personal data to external AI models.
- Gamification synchronization issues when multiple users update shared challenge progress.

Dataset or hardware limitations:

- Image uploads are strictly limited to under 5MB and restricted to JPEG/PNG formats.
- User's device camera quality and lighting negatively affecting food recognition accuracy.
- Strict privacy filters limiting the AI's contextual awareness.
- Chatbot scope explicitly restricted by system prompts to only discuss nutrition and wellness.
- Strict logical boundaries blocking extreme manual inputs (e.g., >5000 kcal or >1000ml water per entry).
- Database bias limiting AI recognition accuracy primarily to the top 100 most common foods.

8. Requirement Traceability Matrix

Requirement ID	**Requirement Description**	**Related Test Cases**
FR-01	Recommended calorie plan generation	TC-01
FR-02	Meal logging via manual search	TC-02
FR-03	Calorie estimation from photo	TC-03
FR-04	Nutritional analysis of text description	TC-04

FR-05	Tracking of health metrics (water, weight, mood)	TC-05
FR-06	Gamification point award	TC-06
FR-07	User profile storage	TC-07
FR-08	Data history storage	TC-08
FR-09	Camera access permissions	TC-09
FR-10	User interaction (commenting/replying)	TC-10
FR-11	Recipe upload and sharing	TC-11
FR-12	User registration and login	TC-12
FR-13	Login streak calculation	TC-13
FR-14	Badge awards for achievements	TC-14
FR-15	Account deletion	TC-15
FR-16	Push notifications	TC-16
FR-17	Editing/deleting logs	TC-17

FR-18	Nutrition summary view	TC-18
FR-19	Adding friends	TC-19
FR-20	Challenge creation	TC-20
FR-21	Challenge invitation response	TC-21
FR-22	Sharing posts within a challenge	TC-22
FR-23	Winner reward	TC-23
FR-24*	NutriCoach chatbot nutrition query and response	TC-24
FR-25*	Chat session history persistence and retrieval	TC-25
FR-26*	Personal data anonymization in chatbot messages	TC-26
FR-27*	Multilingual chatbot interaction (Turkish and English	TC-27

"*" : They were not added to the functional requirement table in the previous report.

9. Conclusion

This test plan provides a structured and systematic approach to validating the developed system. By implementing the defined testing methodologies ranging from unit to user testing we ensure that the system's functionality, performance, and reliability are thoroughly evaluated.

The execution of this plan serves to identify potential errors early, verify that all project requirements are met, and confirm that the software is stable and ready for final deployment. Ultimately, this process guarantees the delivery of a high-quality solution that meets user expectations.