CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: BetaKey: Game Testing Access Portal

Prepared By: Ron Luigi Taghoy

Date of Submission:

Version: 0.0.1

Table of Contents

Contents

1. Introduction

1.1. Purpose

The purpose of this document is to define the functional requirements for "BetaKey", a secure access portal for a closed game testing phase. The system allows testers to register for the program, authenticate their identity, and retrieve their unique beta access keys.

1.2. Scope

The BetaKey system is an authentication application designed to manage the distribution of exclusive game testing access keys. The system encompasses the full lifecycle of user access, starting from the registration of new tester accounts. It includes a secure login mechanism to authenticate users. Once authenticated, the system grants access to a protected mobile dashboard where users can view their unique beta access key. The system handles session management locally on the device, ensuring the key remains accessible only to the authorized user.

1.3. Definitions, Acronyms, and Abbreviations

Definitions, Acronyms, and Abbreviations

- SRS: Functional Requirements Specification
- JWT: JSON Web Token (used for secure, stateless session management)
- Beta Key: A unique alphanumeric code granting access to the game software.
- Tester: An authenticated user who has registered for the program.
- Guest: An unauthenticated user who can only access public pages.

2. Overall Description

2.1. System Perspective

BetaKey acts as a standalone Android application. It serves as the portable "digital wallet" for a game tester's credentials. It communicates with a backend API to verify credentials and retrieve data, displaying the result on the mobile interface.

2.2. User Classes and Characteristics

- Guest User: A user who has downloaded the app but not logged in. They can only access the Login Activity and Registration Activity.
- Authenticated Tester: A registered user who has successfully logged in. They have access to the Main Activity (Dashboard) to view their Beta Key.

2.3. Operating Environment

- Client-Side: Android Smartphone running Android 10.0 (Q) or higher.
- Development Language: Kotlin.
- Server-Side: Backend API (Spring Boot/Java) for data processing.
- Database: MySQL or PostgreSQL.

2.4. Assumptions and Dependencies

- It is assumed that the user has a valid email address for registration.
- The system assumes the underlying database server is operational and accessible.

3. System Features and Functional Requirements

Describe each major feature of the system and its functional requirements.

3.1. Feature 1:

Description: Tester Registration
Functional Requirements:
- The app shall provide a Registration Screen requiring Username, Email, and Password
- The app shall validate the email format before sending data to the server
- Upon success, the app shall automatically navigate the user to the Login Screen

3.2. Feature 2:

Description: A protected screen that displays the user's unique key
Functional Requirements:
- The app shall restrict access to the Dashboard Screen to users with a valid stored token
- The Dashboard shall display the user's username and Beta Key in large, readable text
- The app shall provide a Logout button that clears the stored token and returns the user to the Login Screen

4. Non-Functional Requirements

- Security: User passwords must never be stored in plain text; they must be hashed using industry-standard algorithms (e.g., BCrypt).
- Performance: The login process should complete within 2 seconds under normal load.
- Usability: The user interface must be intuitive, requiring no more than 3 clicks to register and view a key.
- Reliability: The system should handle database connection timeouts gracefully by showing a user-friendly error message rather than crashing,

5. System Models (Diagrams)

Insert the necessary diagrams for the system:

5.1. ERD

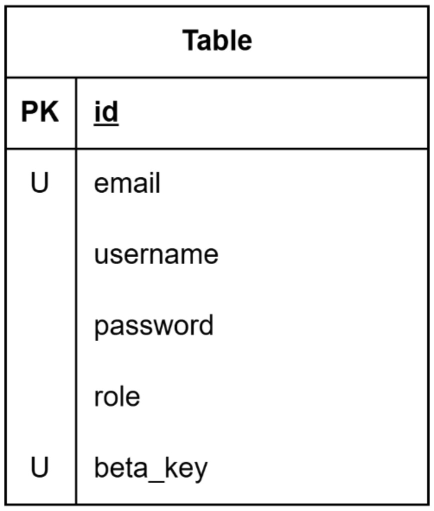
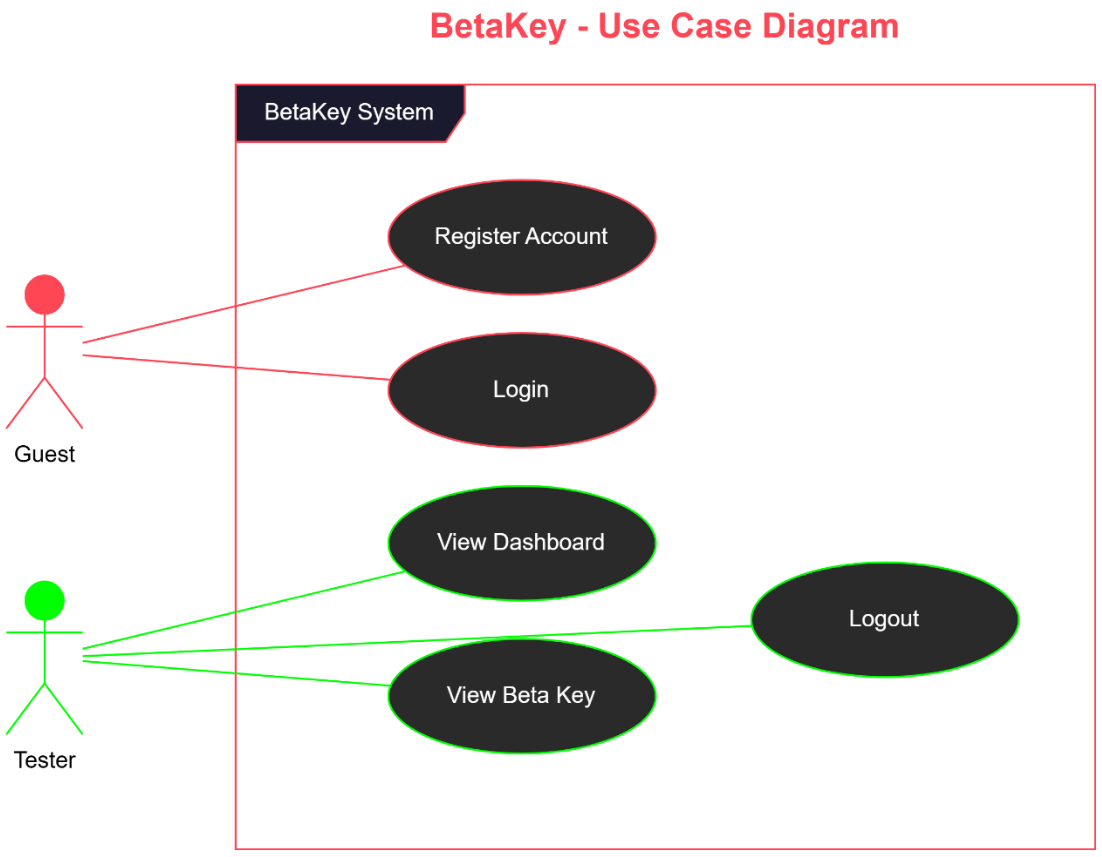
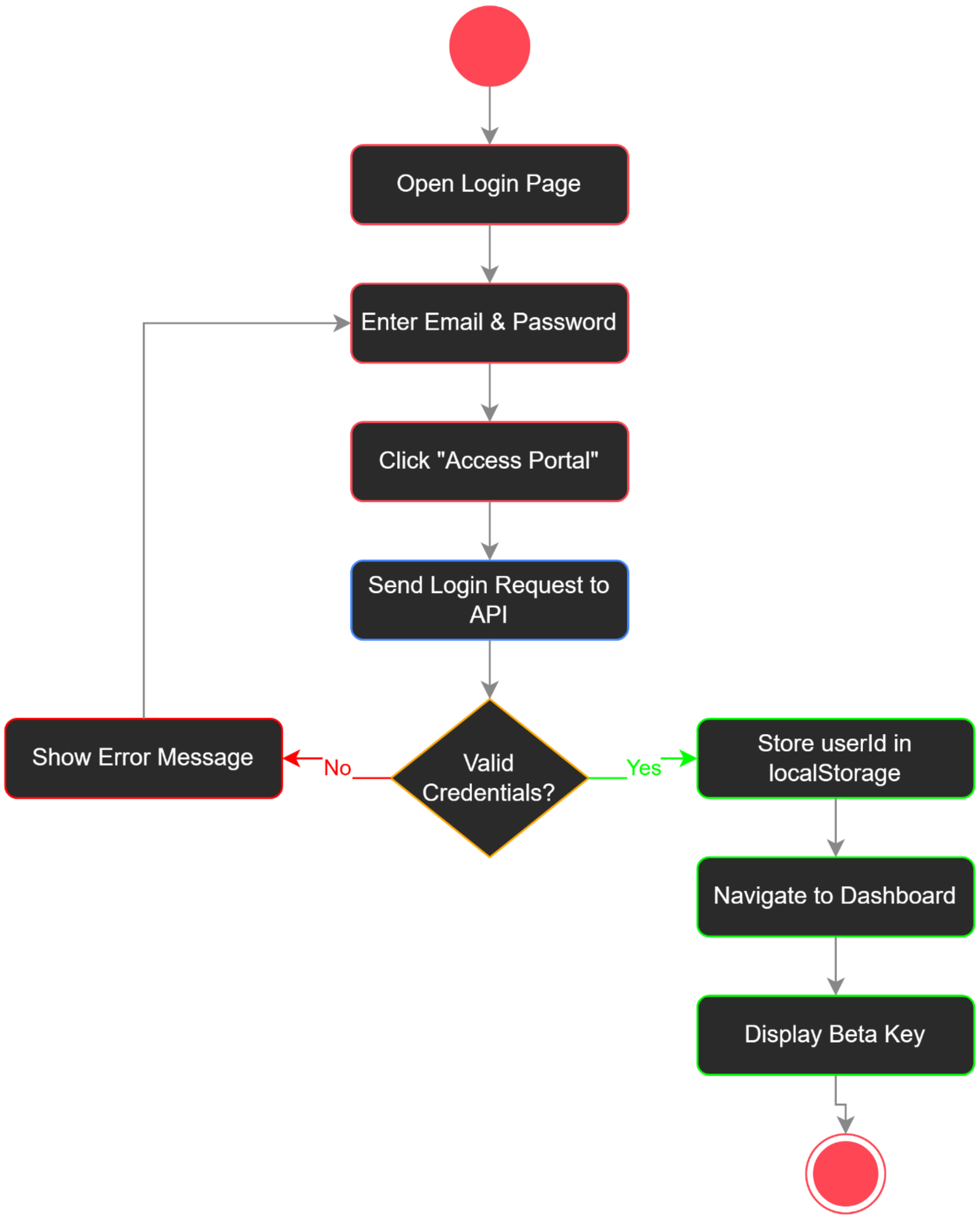


Table	
PK	**<u>id</u>**
U	email
	username
	password
	role
U	beta_key

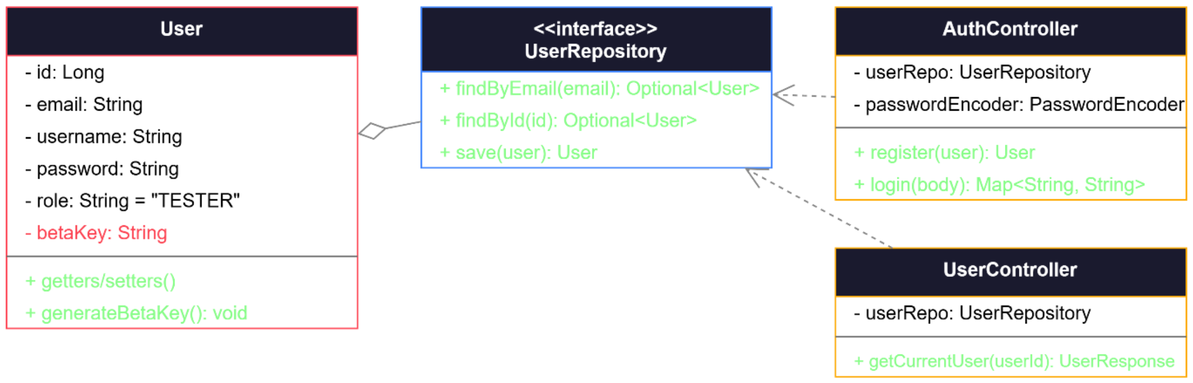
5.2. Use Case Diagram



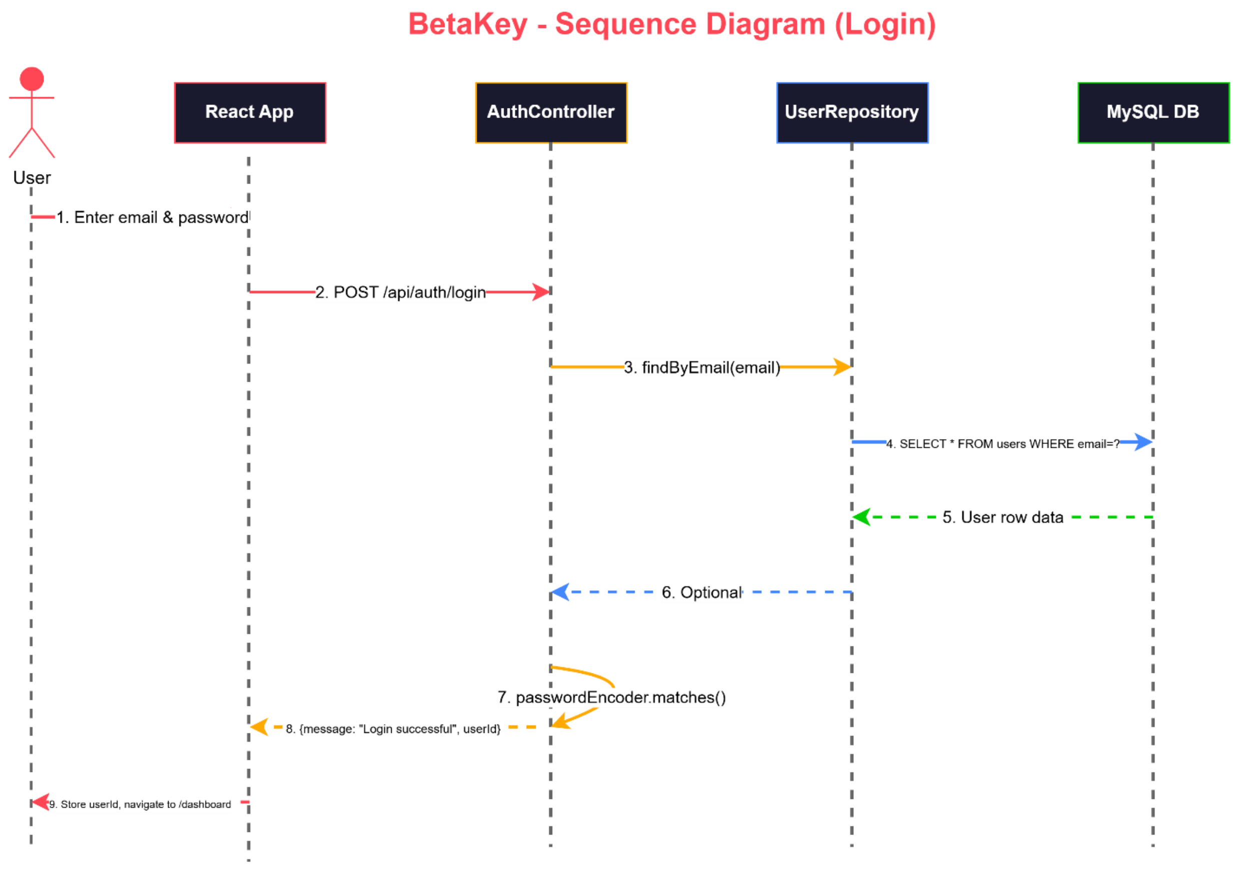
5.3. Activity Diagram



5.4. Class Diagram

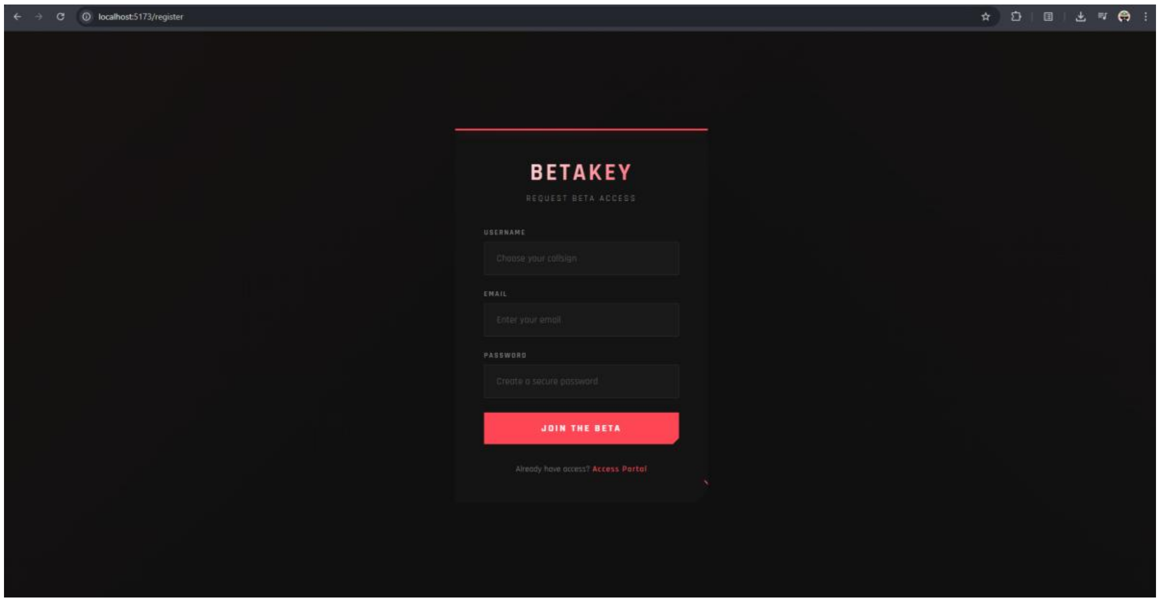

User
- id: Long
- email: String
- username: String
- password: String
- role: String = "TESTER"
- betaKey: String
+ getters/setters()
+ generateBetaKey(): void
<<interface>>
UserRepository
+ findByEmail(email): Optional<User>
+ findById(id): Optional<User>
+ save(user): User
AuthController
- userRepo: UserRepository
- passwordEncoder: PasswordEncoder
+ register(user): User
+ login(body): Map<String, String>
UserController
- userRepo: UserRepository
+ getCurrentUser(userId): UserResponse

5.5. Sequence Diagram

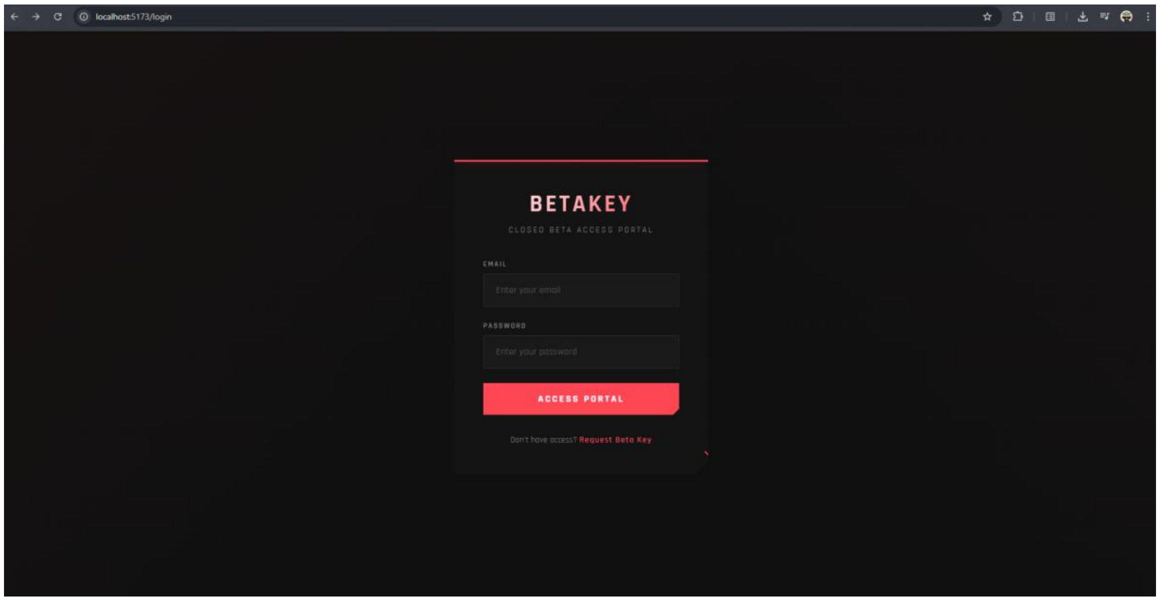

BetaKey - Sequence Diagram (Login)
User
React App
AuthController
UserRepository
MySQL DB
1. Enter email & password
2. POST /api/auth/login
3. findByEmail(email)
4. SELECT * FROM users WHERE email=?
5. User row data
6. Optional
7. passwordEncoder.matches()
8. {message: "Login successful", userId}
9. Store userId, navigate to /dashboard

6. Appendices

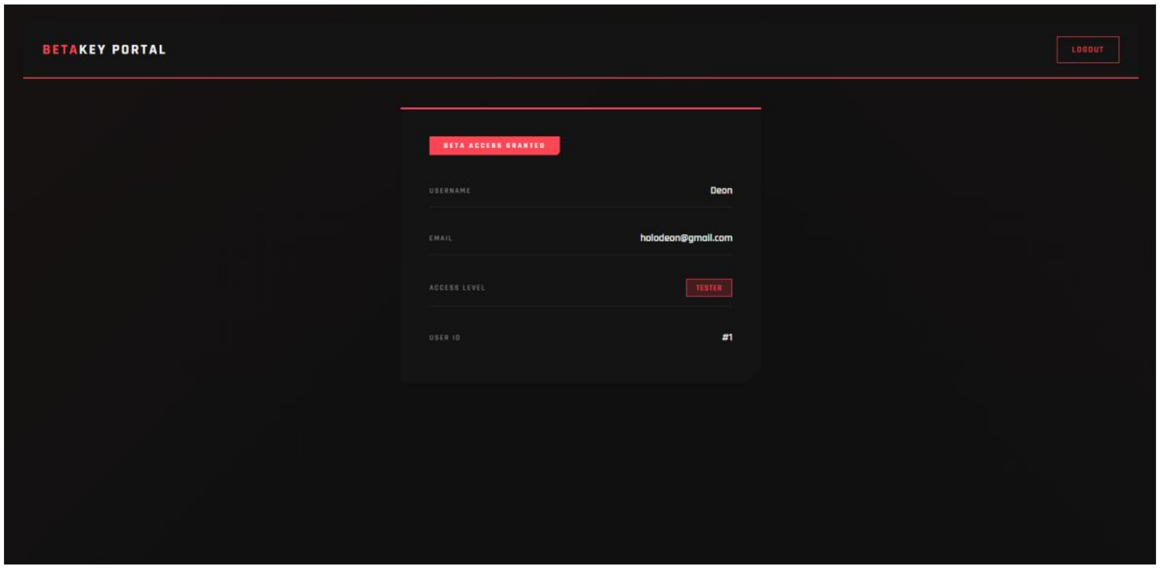
Register:



Login:

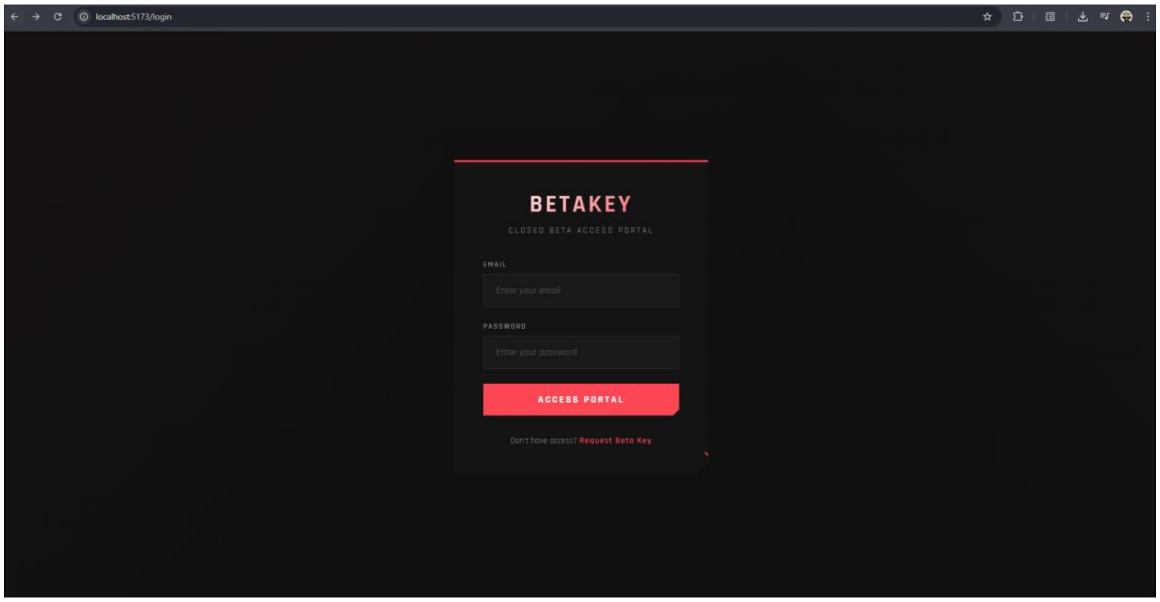


Dashboard/Profile:



USERNAME	Deon
EMAIL	holodeon@gmail.com
ACCESS LEVEL	TESTER
USER ID	#1

Logout:



Don't have access? Request Beta Key