CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: BrainBox - A Centralized Learning and Knowledge Tool

Prepared By: Joana Carla Gako

Date of Submission: January 30, 2026

Version: 1

Table of Contents

1. Introduction

1.1. Purpose

Describe the purpose of the system and the intended audience of this document.

1.2. Scope

Describe what the system will do and its boundaries.

1.3. Definitions, Acronyms, and Abbreviations

List and define important terms used in this document.

2. Overall Description

2.1. System Perspective

Describe how the system fits into a larger context or environment.

2.2. User Classes and Characteristics

Identify the different types of users and their characteristics.

2.3. Operating Environment

Specify the hardware, software, and tools required to operate the system.

2.4. Assumptions and Dependencies

List any assumptions and external dependencies that may affect the system.

3. System Features and Functional Requirements

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

3.1. Feature 1:

Description:

Functional Requirements:

-
-
-

3.2. Feature 2:

Description:

Functional Requirements:

-
-
-

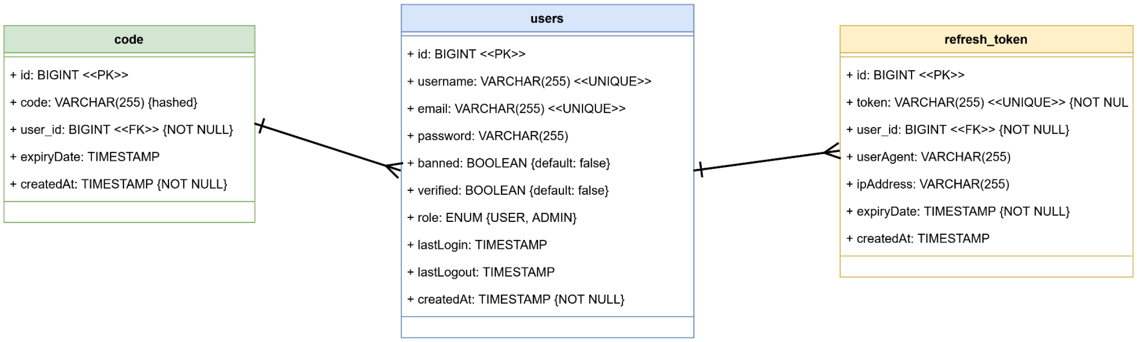
4. Non-Functional Requirements

Specify system quality attributes such as performance, security, usability, reliability, etc.

5. System Models (Diagrams)

Insert the necessary diagrams for the system:

5.1. ERD



code

+ id: BIGINT <<PK>>
+ code: VARCHAR(255) {hashed}
+ user_id: BIGINT <<FK>> {NOT NULL}
+ expiryDate: TIMESTAMP
+ createdAt: TIMESTAMP {NOT NULL}

users

+ id: BIGINT <<PK>>
+ username: VARCHAR(255) <<UNIQUE>>
+ email: VARCHAR(255) <<UNIQUE>>
+ password: VARCHAR(255)
+ banned: BOOLEAN {default: false}
+ verified: BOOLEAN {default: false}
+ role: ENUM {USER, ADMIN}
+ lastLogin: TIMESTAMP
+ lastLogout: TIMESTAMP
+ createdAt: TIMESTAMP {NOT NULL}

refresh_token

+ id: BIGINT <<PK>>
+ token: VARCHAR(255) <<UNIQUE>> {NOT NUL
+ user_id: BIGINT <<FK>> {NOT NULL}
+ userAgent: VARCHAR(255)
+ ipAddress: VARCHAR(255)
+ expiryDate: TIMESTAMP {NOT NULL}
+ createdAt: TIMESTAMP

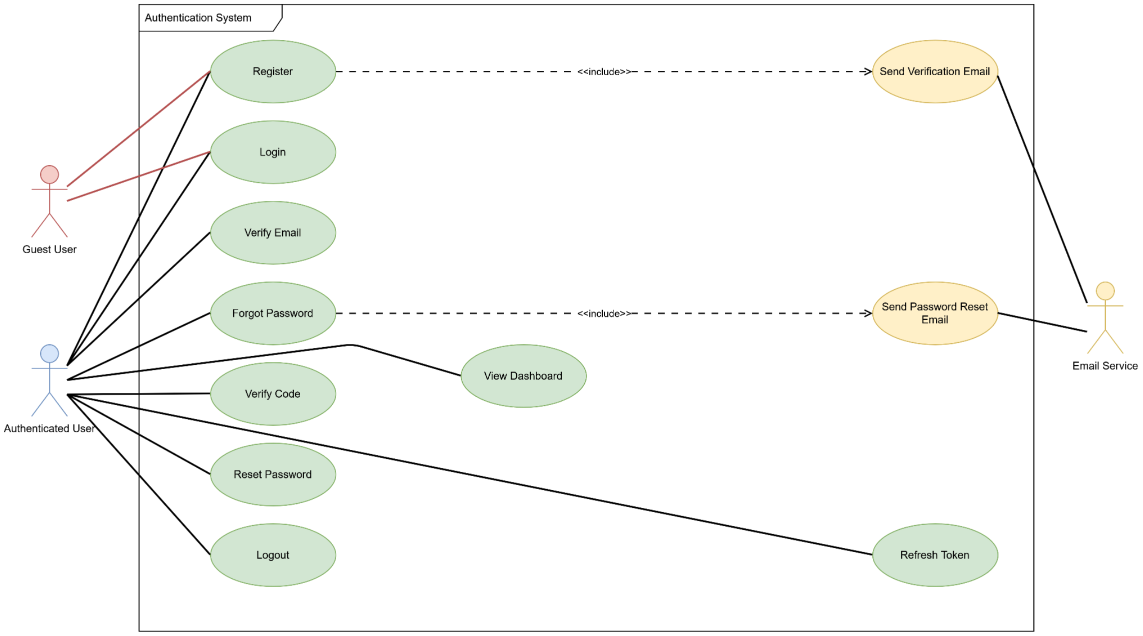
Entity Relationships

• code.user_id → users.id (One-to-One)
• refresh_token.user_id → users.id (One-to-Many)

Constraints

• users.username is UNIQUE
• users.email is UNIQUE
• refresh_token.token is UNIQUE
• Each code belongs to exactly one user
• Each user can have multiple refresh tokens

5.2. Use Case Diagram

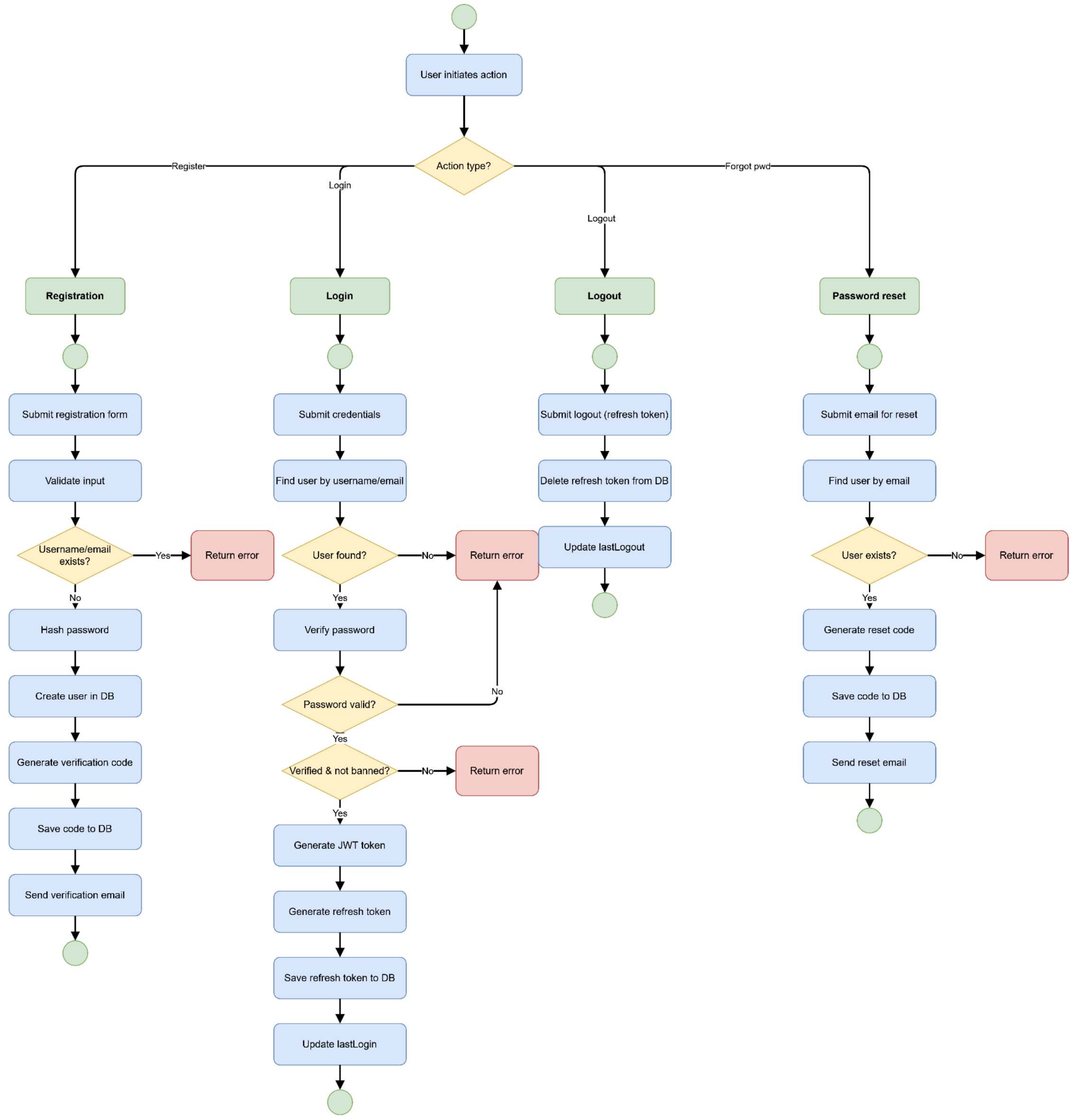


Use Case Summary

Actors:
• Guest User - Can register and login only
• Authenticated User - Has full system access
• Email Service - External system for notifications

Main Flows:
• Guest users can register and login
• Dashboard is protected - only authenticated users can access
• User registration triggers verification email
• Password reset flow requires email verification
• Token refresh extends user sessions

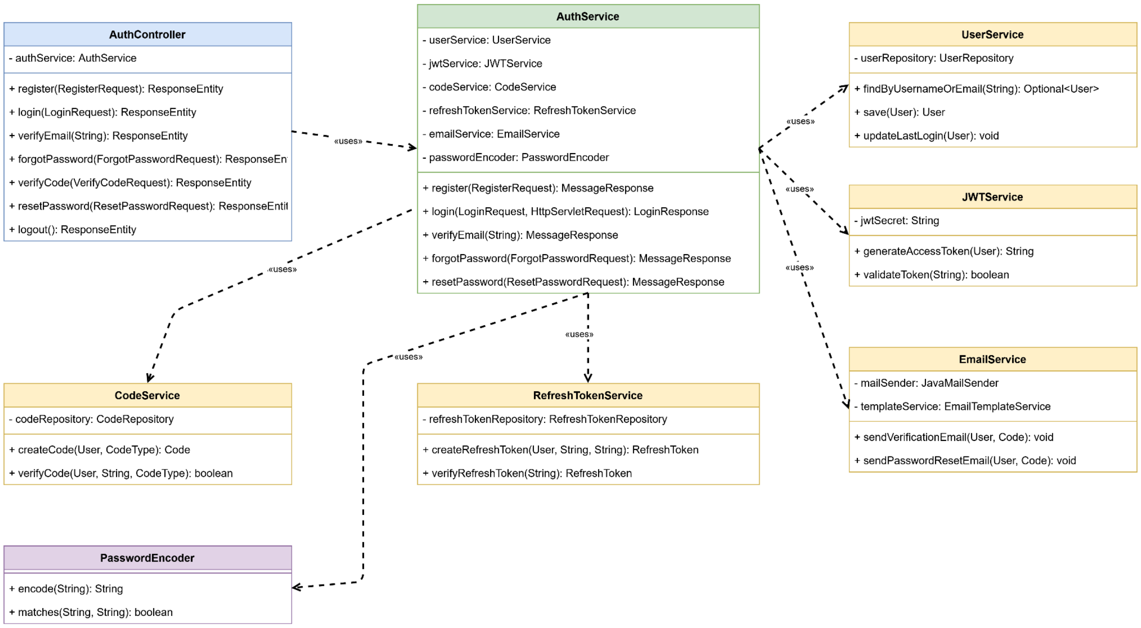
5.3. Activity Diagram



Summary

- **Registration:** Submit form → validate → check username/email → hash password → create user → send verification email
- **Login:** Submit credentials → find user → verify password → check verified/not banned → issue tokens → update lastLogin
- **Logout:** Submit refresh token → delete token from DB → update lastLogout
- **Password reset:** Submit email → find user → generate/save code → send reset email

5.4. Class Diagram


AuthController
- authService: AuthService
+ register(RegisterRequest): ResponseEntity
+ login(LoginRequest): ResponseEntity
+ verifyEmail(String): ResponseEntity
+ forgotPassword(ForgotPasswordRequest): ResponseEn
+ verifyCode(VerifyCodeRequest): ResponseEntity
+ resetPassword(ResetPasswordRequest): ResponseEntit
+ logout(): ResponseEntity
«uses»
AuthService
- userService: UserService
- jwtService: JWTService
- codeService: CodeService
- refreshTokenService: RefreshTokenService
- emailService: EmailService
- passwordEncoder: PasswordEncoder
+ register(RegisterRequest): MessageResponse
+ login(LoginRequest, HttpServletRequest): LoginResponse
+ verifyEmail(String): MessageResponse
+ forgotPassword(ForgotPasswordRequest): MessageResponse
+ resetPassword(ResetPasswordRequest): MessageResponse
«uses»
UserService
- userRepository: UserRepository
+ findByUsernameOrEmail(String): Optional<User>
+ save(User): User
+ updateLastLogin(User): void
«uses»
JWTService
- jwtSecret: String
+ generateAccessToken(User): String
+ validateToken(String): boolean
«uses»
EmailService
- mailSender: JavaMailSender
- templateService: EmailTemplateService
+ sendVerificationEmail(User, Code): void
+ sendPasswordResetEmail(User, Code): void
«uses»
CodeService
- codeRepository: CodeRepository
+ createCode(User, CodeType): Code
+ verifyCode(User, String, CodeType): boolean
«uses»
RefreshTokenService
- refreshTokenRepository: RefreshTokenRepository
+ createRefreshToken(User, String, String): RefreshToken
+ verifyRefreshToken(String): RefreshToken
«uses»
PasswordEncoder
+ encode(String): String
+ matches(String, String): boolean

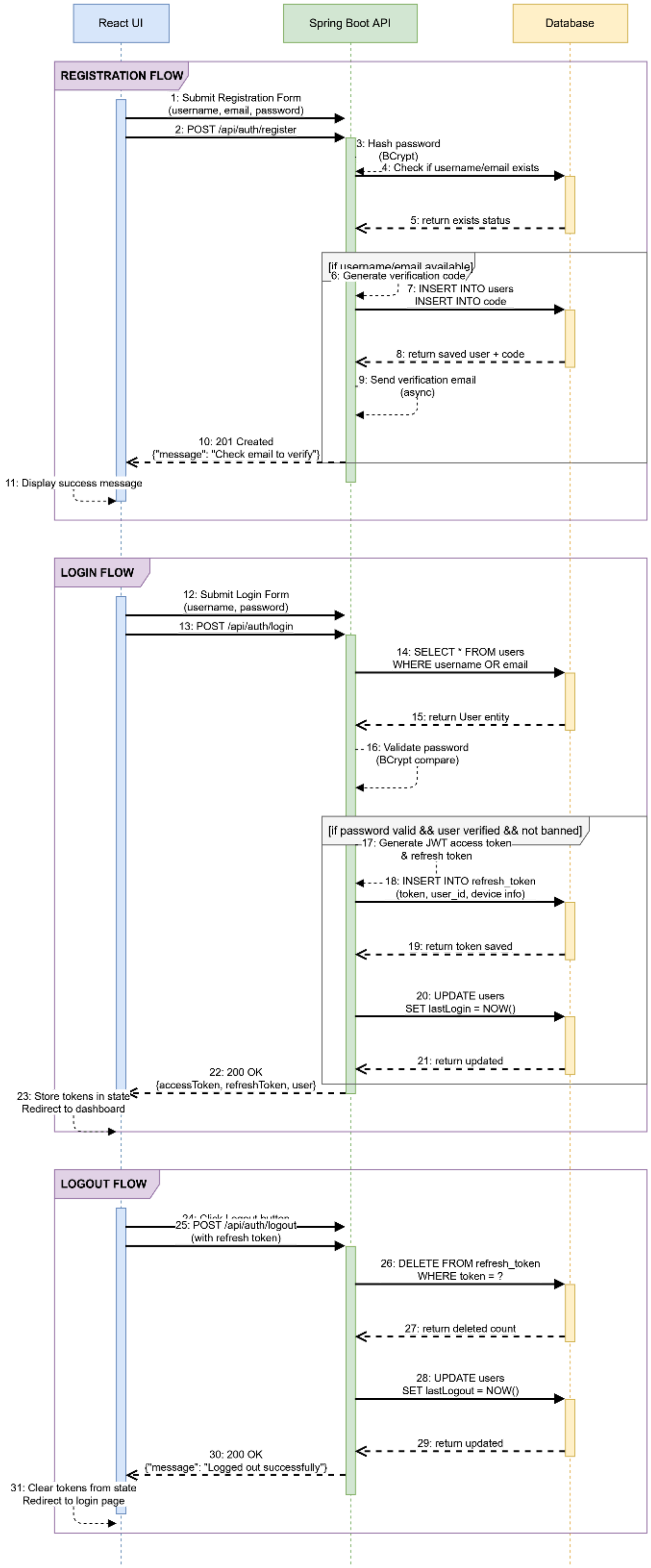
Service Layer Architecture

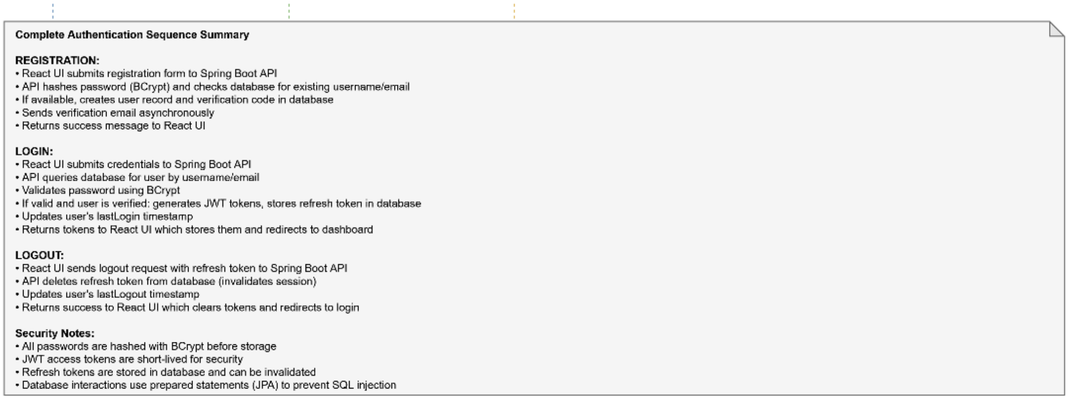
The AuthController delegates business logic to AuthService, which coordinates multiple specialized services:

- UserService: User data management
- JWTService: Access token generation and validation
- CodeService: Verification code handling
- RefreshTokenService: Refresh token lifecycle
- EmailService: Email delivery
- PasswordEncoder: Password hashing and verification

All services use Spring's dependency injection with @Service and @RequiredArgsConstructor annotations.

5.5. Sequence Diagram





Complete Authentication Sequence Summary

REGISTRATION:
- React UI submits registration form to Spring Boot API
- API hashes password (BCrypt) and checks database for existing username/email
- If available, creates user record and verification code in database
- Sends verification email asynchronously
- Returns success message to React UI

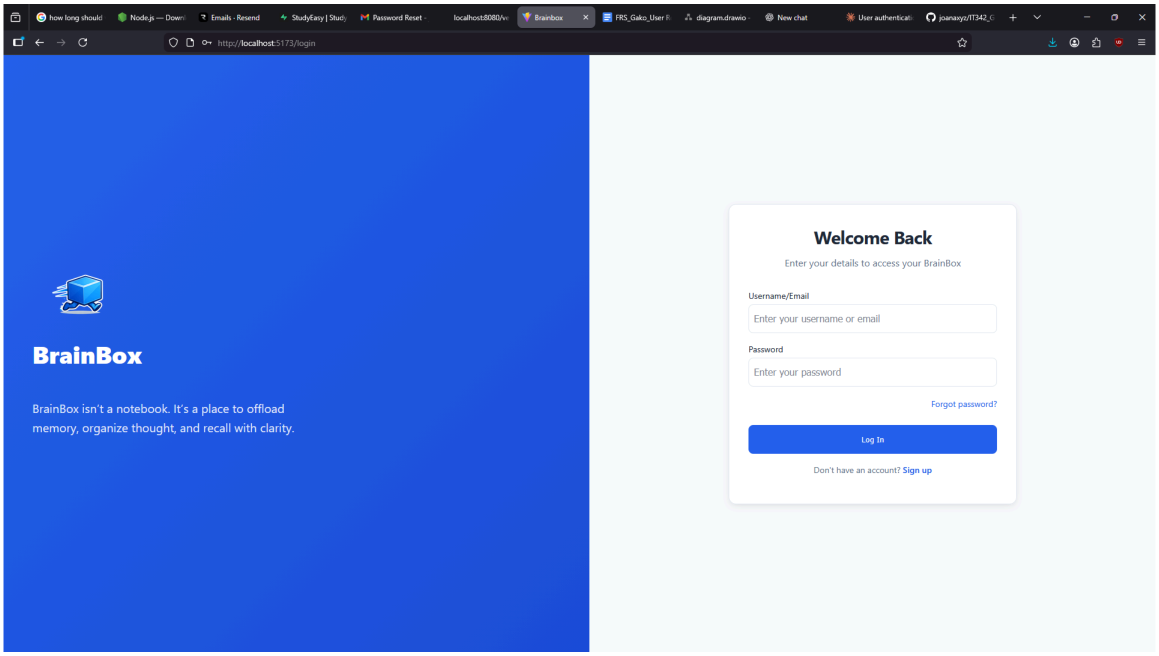
LOGIN:
- React UI submits credentials to Spring Boot API
- API queries database for user by username/email
- Validates password using BCrypt
- If valid and user is verified: generates JWT tokens, stores refresh token in database
- Updates user's lastLogin timestamp
- Returns tokens to React UI which stores them and redirects to dashboard

LOGOUT:
- React UI sends logout request with refresh token to Spring Boot API
- API deletes refresh token from database (invalidates session)
- Updates user's lastLogout timestamp
- Returns success to React UI which clears tokens and redirects to login

Security Notes:
- All passwords are hashed with BCrypt before storage
- JWT access tokens are short-lived for security
- Refresh tokens are stored in database and can be invalidated
- Database interactions use prepared statements (JPA) to prevent SQL injection

6. Appendices

Include any additional information, references, or support materials.



BrainBox

BrainBox isn't a notebook. It's a place to offload memory, organize thought, and recall with clarity.

Welcome Back

Enter your details to access your BrainBox

Username/Email

Enter your username or email

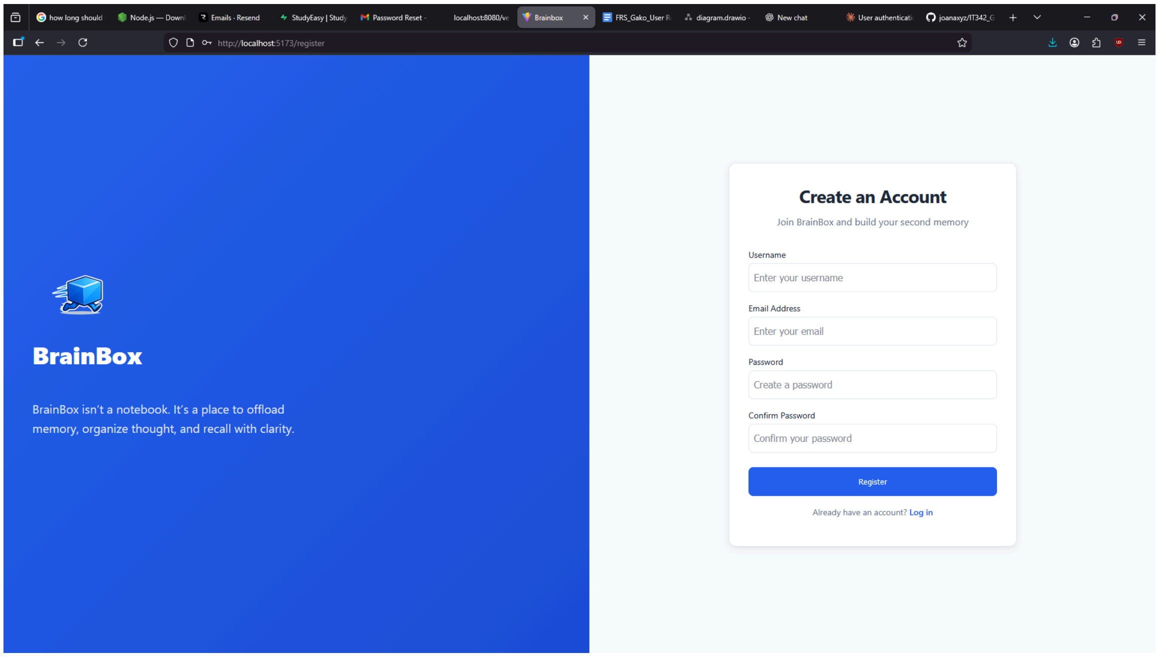
Password

Enter your password

Forgot password?

Log In

Don't have an account? Sign up



BrainBox

BrainBox isn't a notebook. It's a place to offload memory, organize thought, and recall with clarity.

Create an Account

Join BrainBox and build your second memory

Username

Enter your username

Email Address

Enter your email

Password

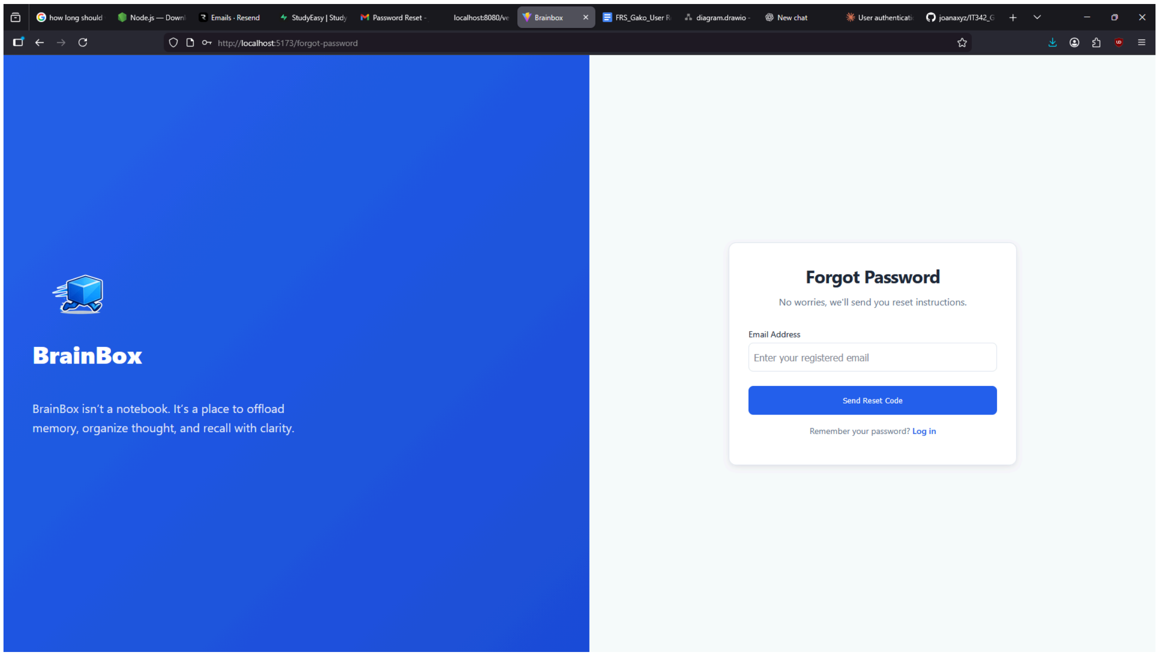
Create a password

Confirm Password

Confirm your password

Register

Already have an account? Log in



BrainBox

BrainBox isn't a notebook. It's a place to offload memory, organize thought, and recall with clarity.

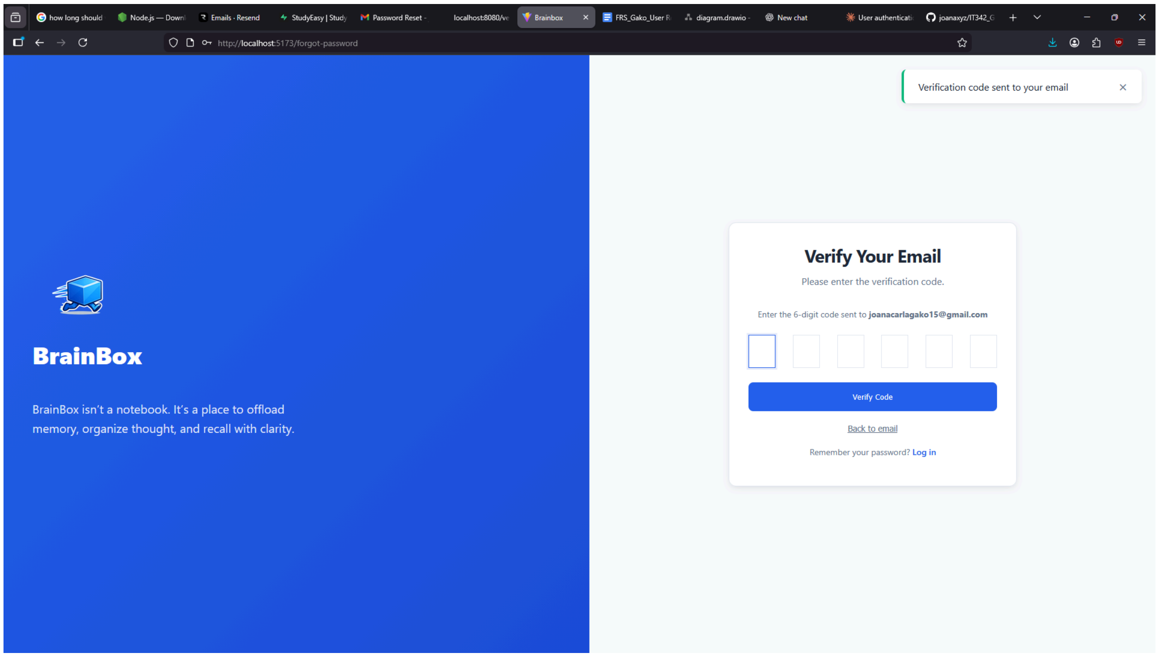
Forgot Password

No worries, we'll send you reset instructions.

Email Address

Enter your registered email

Remember your password? Log in



Verification code sent to your email

BrainBox

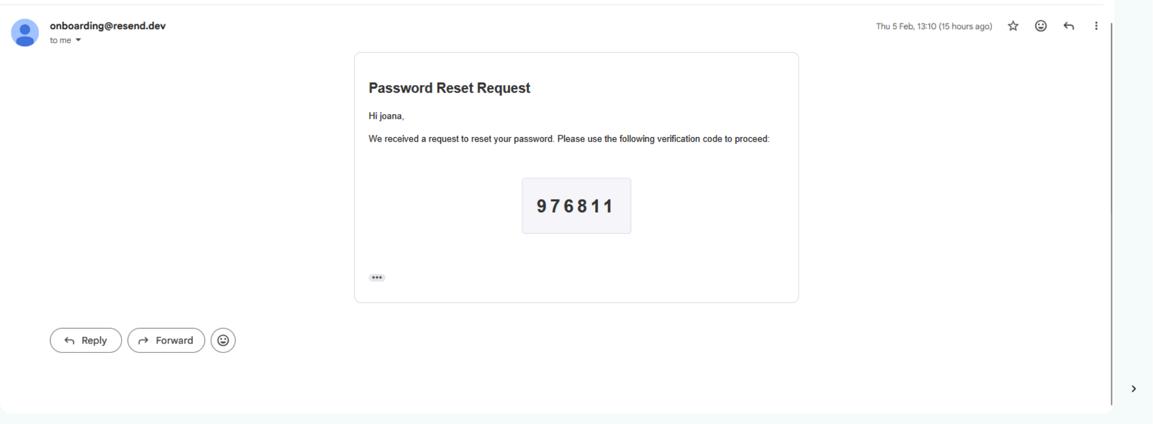
BrainBox isn't a notebook. It's a place to offload memory, organize thought, and recall with clarity.

Verify Your Email

Please enter the verification code.

Enter the 6-digit code sent to **joanacarlagako15@gmail.com**

Remember your password? Log in



onboarding@resend.dev
to me

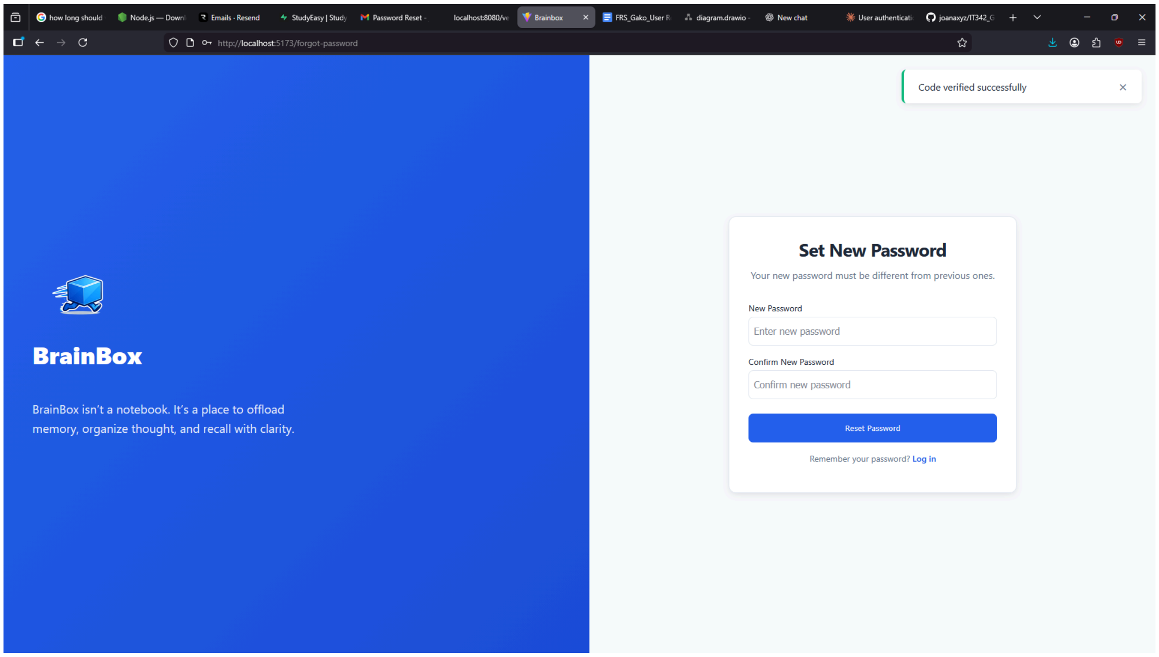
Thu 5 Feb, 13:10 (15 hours ago)

Password Reset Request

Hi joana,

We received a request to reset your password. Please use the following verification code to proceed:

976811



http://localhost:5173/forgot-password

Code verified successfully

BrainBox

BrainBox isn't a notebook. It's a place to offload memory, organize thought, and recall with clarity.

Set New Password

Your new password must be different from previous ones.

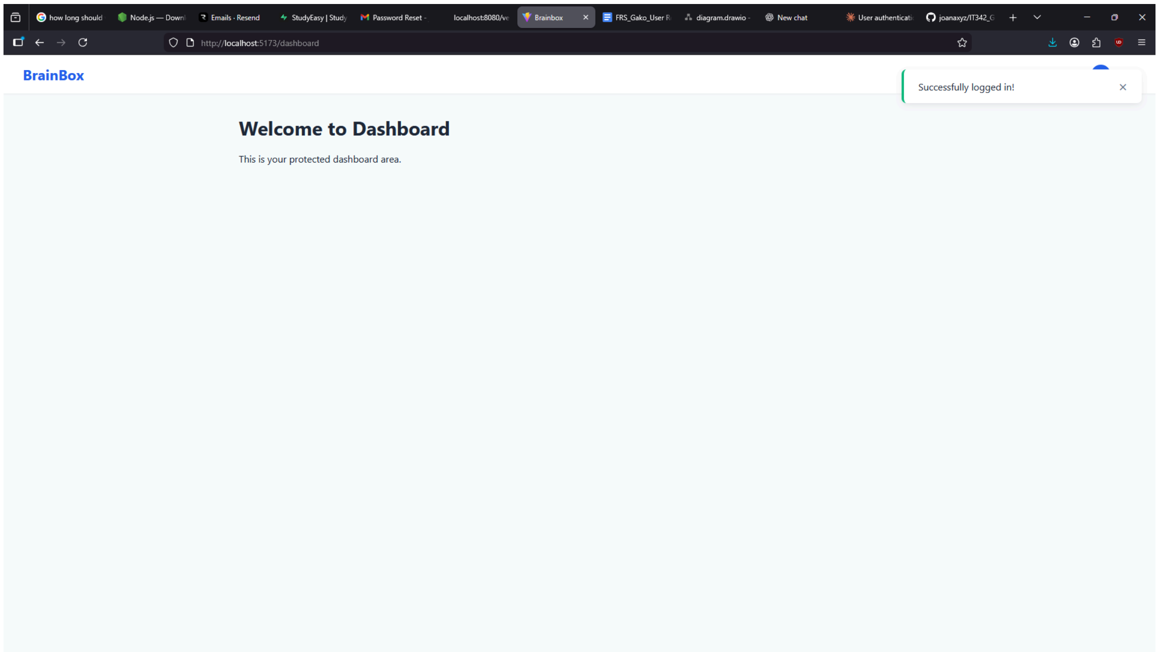
New Password

Enter new password

Confirm New Password

Confirm new password

Remember your password? Log in

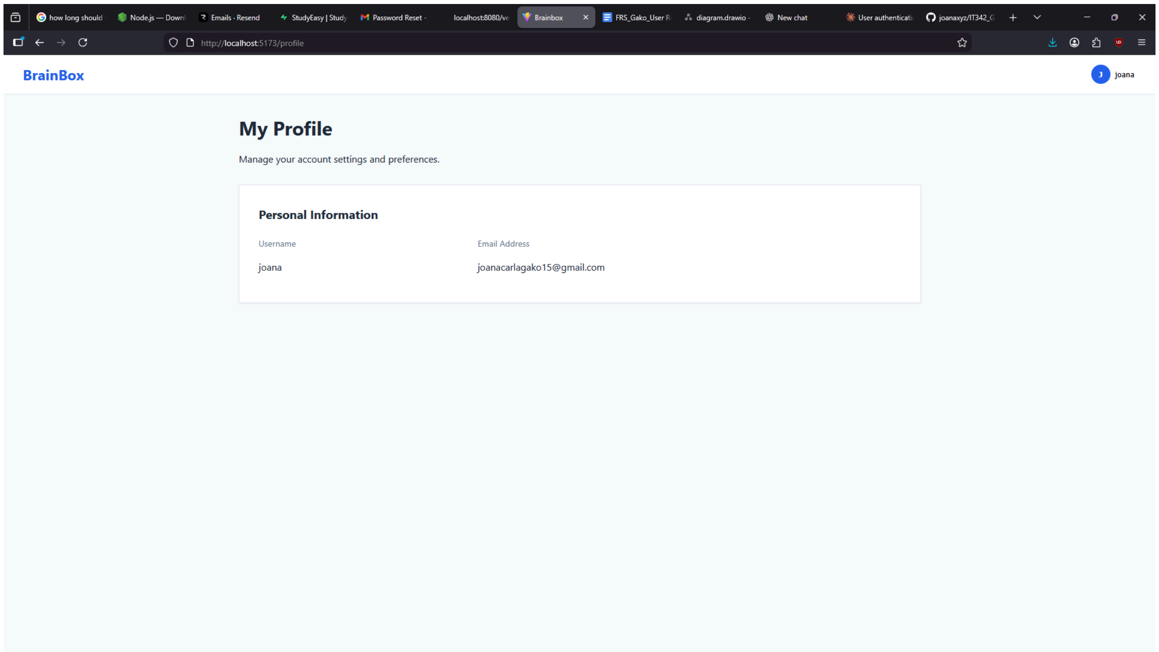


BrainBox

Successfully logged in!

Welcome to Dashboard

This is your protected dashboard area.

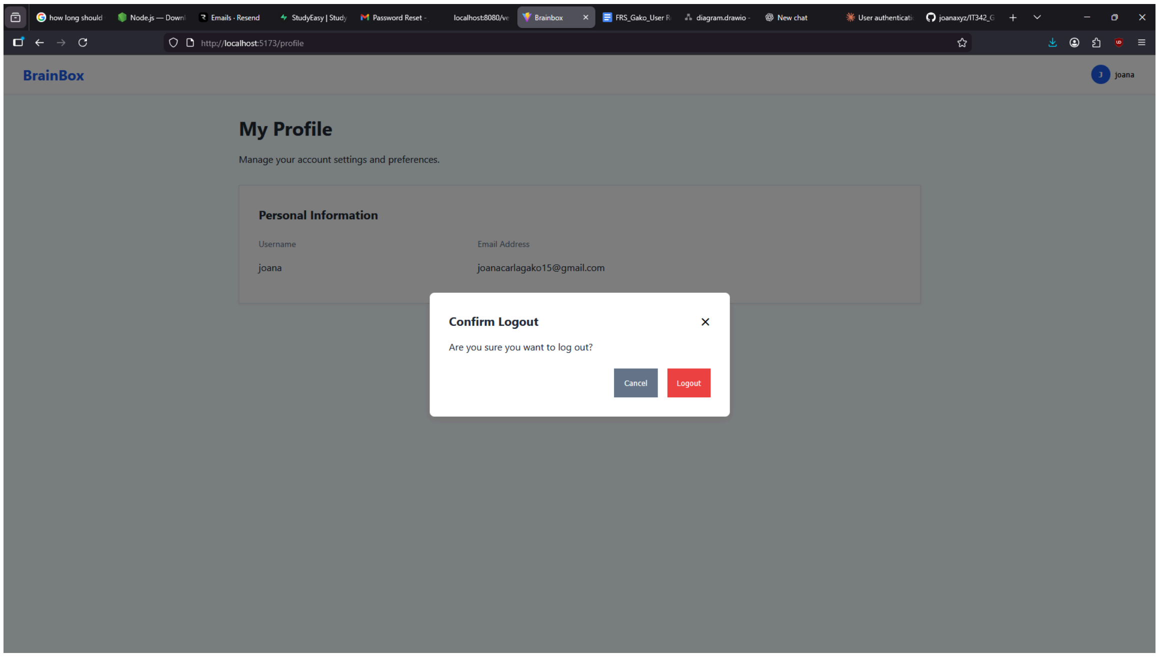


My Profile

Manage your account settings and preferences.

Personal Information

Username	Email Address
joana	joanacarlagako15@gmail.com



My Profile

Manage your account settings and preferences.

Personal Information

Username	Email Address
joana	joanacarlagako15@gmail.com

Confirm Logout

Are you sure you want to log out?