CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G4
System Integration and Architecture

System Design Document (SDD)

Project Title: PeerTayo

Prepared By: Camoro, Mark Anton Ledesma

Version: 1.0

Date: 02 / 08 / 2026
Status: Draft

REVISION HISTORY TABLE

Version	Date	Author	Changes Made	Status
0.1	02/18/2026	Mark Anton L. Camoro	Initial draft	Draft
0.2	[Date]	[Your Name]	Added API specifications	Review
0.3	[Date]	[Your Name]	Updated database design	Review
0.4	[Date]	[Your Name]	Added UI/UX designs	Review
0.5	[Date]	[Your Name]	Incorporated feedback	Revised
0.6	[Date]	[Your Name]	Final review and corrections	Final
1	[Date]	[Your Name]	Baseline version for development	Approved

TABLE OF CONTENTS

Contents

EXECUTIVE SUMMARY..... 4

1.0 INTRODUCTION.....6

2.0 FUNCTIONAL REQUIREMENTS SPECIFICATION.....6

3.0 NON-FUNCTIONAL REQUIREMENTS.....20

4.0 SYSTEM ARCHITECTURE.....24

5.0 API CONTRACT & COMMUNICATION.....25

Authentication Endpoints..... 26

User Registration

User Login

Google OAuth Login

Get Current User

User Logout

Evaluation Endpoints.....30

Create Evaluation

Get Pending Evaluations

Submit Evaluation

Get My Results

Get Created Evaluations

Get Evaluation Results (Facilitator)

Notification Endpoints.....36

Get Notifications

Mark Notification as Read

Error Handling.....38

6.0 DATABASE DESIGN.....43

7.0 UI/UX DESIGN.....48

8.0 PLAN.....71

EXECUTIVE SUMMARY

1.1 Project Overview

PeerTayo Evaluation System is a web and mobile-based platform designed to support structured peer performance assessments among registered users. It allows users to create standardized evaluation forms, assign evaluators and evaluatees, submit rating-based evaluations, and view summarized performance results. The system also enables users to monitor and manage the evaluations they create, ensuring an organized and transparent evaluation workflow.

The platform uses a multi-role architecture where every registered user is automatically assigned the Respondent role, while users who create evaluations are granted the Facilitator role—allowing a single user to hold both roles simultaneously. Security is enforced through JWT-based authentication, BCrypt password hashing, role-based access control, and protected API routes. With both web and mobile interfaces, PeerTayo ensures accessible and secure performance evaluation across devices.

1.2 Objectives

1. Develop a secure peer evaluation system for web and mobile platforms
2. Implement JWT-based authentication
3. Encrypt passwords using BCrypt
4. Support Google OAuth login integration
5. Implement role-based authorization (Respondent and Facilitator)
6. Allow creation and distribution of evaluation forms
7. Enable selection of evaluators and evaluatees
8. Provide structured 10-criterion rating-scale evaluations
9. Display summarized evaluation results with graphical representation
10. Implement in-app notifications
11. Implement SMTP email notifications
12. Integrate an external public API into the dashboard
13. Maintain a clean, responsive UI for both web and mobile

1.3 Scope

Included Features:

- Web application
- Mobile application
- User registration and login
- JWT authentication
- BCrypt password encryption
- Google OAuth login integration
- Multi-role system (Respondent + Facilitator)
- Role-based route protection
- Fixed 10-criterion standardized criteria
- Creation of evaluation forms
- Assignment of evaluators and evaluatees
- Evaluation deadline setting
- Dashboard displaying pending evaluations
- In-app notification system
- Email notification system (SMTP)
- Submission of evaluations
- Viewing personal evaluation results
- Viewing results of created evaluations
- Tracking completed evaluations
- Graphical statistics per criteria
- Optional end-of-form feedback comment
- External API integration displayed in dashboard

Excluded Features:

- Customizable criteria
- Per-criteria comment fields

- Real-time push notifications
- Advanced analytics module
- Role-based administrative panel
- Group or team management module
- File upload functionality
- Role-based administration

1.0 INTRODUCTION

1.1 Purpose

This document serves as the Software Design Description (SDD) for the PeerTayo Evaluation System. It outlines the system requirements, functional specifications, user workflows, and design decisions that will guide development. The document ensures that the system remains simple, practical, and achievable for solo implementation while fulfilling core peer evaluation needs.

2.0 FUNCTIONAL REQUIREMENTS SPECIFICATION

2.1 Project Overview

Project Name: PeerTayo
Domain: Education / Performance Evaluation
Primary Users: Students, Teachers, Team Members, Organizations

Problem Statement: Many institutions and groups require structured peer evaluation but lack a simple system to manage the process digitally. Existing tools are either too complex or too generic.

Solution: A lightweight evaluation platform that allows users to evaluate each other using a standardized rating system with easy result tracking.

2.2 Core User Journey

1: User Registration and Authentication

1. User opens web or mobile app
2. User registers with name, email, and password
3. Password is hashed using BCrypt
4. User logs in using:
    - Email and password OR
    - Google OAuth
5. System generates JWT
6. User is redirected to Dashboard

Default role: RESPONDENT

Journey 2: Creating an Evaluation (Role Expansion)

1. User logs in
2. Navigates to “Create Evaluation”
3. System checks role:
    - If user does not have Facilitator role → system assigns Facilitator role
4. User sees the pre-made criterion for rating
5. User selects:
    - Evaluatees
    - Evaluators
    - Deadline
6. System creates evaluation record
7. Assigned evaluators:
    - See in-app notification
    - Receive SMTP email notification

User now has:
Respondent + Facilitator roles.

Journey 3: Completing an Evaluation

1. User logs in (web or mobile)
2. Dashboard shows pending evaluations

3. User taps/clicks “View”
4. Evaluation form opens
5. User answers:
    - 10 rating criteria (scale 1–5)
    - Optional final comment
6. User submits
7. System:
    - Saves responses
    - Marks evaluation as completed
    - Updates statistics

Journey 4: Viewing Personal Results

1. User opens “My Results”
2. System aggregates responses about user
3. Displays:
    - Average per criteria
    - Overall average score
    - Graphical chart
    - Anonymous feedback comments

Journey 5: Monitoring Created Evaluations

1. User opens “Forms Created”
2. System displays:
    - List of evaluations created
    - Submission progress
3. User selects evaluation
4. System displays:
    - Per-evaluatee results
    - Per-criteria averages
    - Overall statistics

Journey 6: External API Integration

1. User opens dashboard
2. System fetches data from public API
3. Displays widget (e.g., motivational quote or productivity tip)

2.3 Feature List (MoSCoW)

MUST HAVE

1. Web and Mobile platform support
2. JWT authentication
3. BCrypt password encryption
4. Google OAuth login
5. Multi-role system
6. Role-based access restriction
7. Creation of evaluation forms
8. Selection of evaluators and evaluatees
9. Fixed 10-criteria rating scale
10. Deadline setting
11. Pending evaluations dashboard
12. Evaluation submission
13. Viewing personal results
14. Viewing created evaluation results
15. In-app notifications
16. SMTP email notifications
17. Data persistence

SHOULD HAVE

1. Graphical statistics
2. Submission progress tracking
3. Optional final comment
4. External API dashboard widget

COULD HAVE

1. Export results to PDF
2. Filtering/search functionality
3. Theme customization

WON'T HAVE

1. Custom questionnaires
2. Real-time push updates
3. Role-based admin system
4. File uploads
5. Group management module

2.4 Detailed Feature Specifications

Feature: User Authentication

Screens: Registration (Web & Mobile), Login (Web & Mobile)

Fields:

- First Name
- Last Name
- Email
- Password
- Google Login Button

Validation:

- Unique email
- Valid email format
- Password minimum length
- Required fields must not be empty

Functions:

- Create account
- Encrypt password using BCrypt
- Secure login using credential verification
- Generate JWT upon successful login
- Google OAuth login
- Logout (invalidate client session)
- Redirect authenticated users to dashboard
- Send welcome email via SMTP after registration

Feature: Role Management

Location: System-wide (Backend Controlled)

Roles:

- RESPONDENT (default role upon registration)
- FACILITATOR (auto-assigned when creating evaluation)

Rules:

- Users can hold both roles simultaneously
- No role switching
- No admin role exists

Functions:

- Automatically assign Facilitator role when user creates first evaluation
- Restrict Create Evaluation and Forms Created to Facilitator role
- Allow Respondent features to all authenticated users

Feature: Dashboard

Location: Main Landing Page after Login (Web & Mobile)

Displays:

- Greeting message
- Overall performance summary
- Pending evaluations count
- Quick navigation buttons
- Notification indicator
- External API widget (e.g., productivity tip)

Functions:

- Load user-specific data
- Display pending evaluations
- Display summary of received evaluations
- Fetch and display data from external public API
- Navigate to other sections

Feature: Create Evaluation

Screen: Create Evaluation Page (Web & Mobile)

Fields:

- Pre-made Criterion for rating
- Evaluation Title with description
- Radio button for rating (1-5)
- Select Evaluatees (from registered users)
- Select Evaluators (from registered users)
- Deadline date

Validation:

- Title required
- At least one evaluatee selected
- At least one evaluator selected

- Deadline must be valid future date

Functions:

- Automatically assign Facilitator role if not present
- Create evaluation record
- Generate evaluator assignments
- Send in-app notifications
- Send SMTP email notifications to assigned evaluators
- Redirect to Forms Created page

Feature: Pending Evaluations

Location: Dashboard

Displays:

- List of evaluations assigned to logged-in user
- Deadline per evaluation
- Status indicator

Functions:

- Display evaluations where user is evaluator
- “View” button opens evaluation form
- Hide completed evaluations
- Update count after submission

Feature: Evaluation Form

Screen: Evaluation Modal/Page (Web & Mobile)

Content:

- 10 predefined rating-scale criterias
- 1–5 rating scale per criteria
- Optional final feedback comment

Validation:

- All 10 rating criterias must be answered
- Ratings must be within 1–5 range

Process:

- User selects rating for each criteria
- User optionally adds final comment
- User submits evaluation
- System saves responses
- System marks evaluation as completed
- System updates aggregated results
- Remove item from pending list

Feature: My Results

Location: Sidebar → My Results

Purpose:
Show evaluations about the logged-in user (Respondent view)

Displays:

- List of evaluation titles received
- Average rating per criteria
- Overall average score
- Graphical statistics (chart)
- Anonymous optional feedback comments

Functions:

- Aggregate evaluation data
- Compute averages per criteria
- Compute overall score
- Display results in readable format

Feature: Evaluations I Created (Forms Created)

Location: Sidebar → Forms Created

Purpose:
Manage evaluations created by logged-in user (Facilitator view)

Displays:

- List of created evaluations
- Deadline status
- Submission progress (e.g., 3/5 completed)
- View details button

On Detailed View:

- Per-evaluatee results
- Per-criteria averages
- Overall statistics
- Optional comments
- Delete evaluation option

Functions:

- Monitor submission progress
- View evaluation statistics
- Manage created evaluations

Feature: My Completed Evaluations

Location: Sidebar → My Completed Evaluations

Purpose:
Track evaluations submitted by logged-in user

Displays:

- List of completed evaluations
- Date submitted
- Evaluatee name
- View read-only submitted responses

Functions:

- Retrieve user submission history

- Display submitted ratings and comments

Feature: In-App Notifications

Location: Dashboard notification indicator

Triggers:

- User assigned as evaluator
- New evaluation created
- Deadline approaching (optional)

Functions:

- Display unread notification count
- Mark notifications as read
- Navigate to related evaluation

Feature: Email Notifications (SMTP Integration)

Trigger Events:

- Successful registration (Welcome email)
- Assignment as evaluator

Functions:

- Send structured email using SMTP server
- Notify evaluator of new evaluation
- Confirm successful registration

Feature: External API Integration

Location: Dashboard Widget

Purpose:
Enhance dashboard with external informational content

Functions:

- Fetch data from public REST API
- Display external content (e.g., productivity tip)
- Handle API failure gracefully
- Refresh data on dashboard load

2.5 Acceptance Criteria

AC-1 Registration

Given I am a new user
When I enter a valid email address
And enter a strong password
And confirm that the password matches
And click "Register"
Then my account should be created successfully
And my password should be securely encrypted
And I should receive a welcome email
And I should be redirected to the dashboard

AC-2: Successful Login (Email and Password)

Given I have a registered account
When I enter valid login credentials
And click "Login"
Then I should receive a valid JWT token
And I should be authenticated successfully
And I should be redirected to the dashboard

AC-3: Successful Google Login

Given I choose to log in using Google
When Google authentication succeeds
Then my account should be created automatically if it does not exist
And I should be authenticated in the system
And a JWT token should be generated
And I should be redirected to the dashboard

AC-4: Create Evaluation

Given I am logged in as a FACILITATOR
When I create a new evaluation
And set a title, description, and deadline
And assign evaluators and evaluatees
And activate the evaluation
Then the evaluation should be saved successfully
And assigned evaluators should see it in their pending list
And assigned evaluators should receive a notification

AC-5: Complete Evaluation

Given I am assigned as an evaluator
When I open a pending evaluation
And provide ratings (1–5) for all predefined criteria
And submit the evaluation
Then the evaluation should be marked as submitted
And it should be removed from my pending list
And my ratings should be stored successfully

AC-6: View My Results

Given other users have submitted evaluations about me
When I navigate to "My Results"
Then I should see the average rating per criterion
And I should see my overall average score
And I should see any optional comments provided

AC-7: Monitor Created Evaluations

Given I created an evaluation
When I open "Forms Created"
Then I should see the list of my created evaluations
And I should see the submission progress for each
And I should be able to view detailed aggregated results

AC-8: View Completed Evaluations

Given I have submitted evaluations
When I open "My Completed Evaluations"
Then I should see the list of evaluations I completed
And I should see the submission date
And I should be able to view the ratings I provided

AC-9: Dashboard Display

Given I am logged in
When I access the dashboard
Then I should see my pending evaluations
And I should see a summary of my recent activity
And relevant system notifications should be displayed

AC-10: Role-Based Access Control

Given I am logged in as a RESPONDENT
When I attempt to create an evaluation
Then access should be denied

Given I am logged in as a FACILITATOR
When I create an evaluation
Then the action should be permitted

AC-11: Mobile Application Accessibility

Given I access the PeerTayo mobile application
When I log in successfully
Then I should be able to create evaluations
And complete assigned evaluations
And view my results
And receive notifications

3.0 NON-FUNCTIONAL REQUIREMENTS

3.1 Performance Requirements

API Response Time:

- ≤ 2 seconds for 95% of requests under normal usage

Web Page Load Time:

- ≤ 3 seconds on standard broadband connection

Mobile Application Startup:

- ≤ 3 seconds cold start

Concurrent Users:

- Support at least 10–50 concurrent users (suitable for classroom or small institution use)

Database Performance:

- Most queries (evaluation retrieval, results aggregation) complete within 500ms

Scalability Scope:

- Designed for small-to-medium academic environments
- Not intended for enterprise-scale deployment

These targets are feasible using:

- Spring Boot
- MySQL
- Proper indexing
- Simple query design

3.2 Security Requirements

Authentication:

- JWT-based authentication
- Token required for protected endpoints
- /me endpoint returns authenticated user details

Password Security:

- Password hashing using BCrypt (salt rounds = 10–12)
- No plain-text password storage

Authorization:

- Role-based restrictions:
    - RESPONDENT
    - FACILITATOR
- API-level role validation
- UI-level access restriction

Data Protection:

- HTTPS required in production
- SQL injection prevention via JPA parameterized queries
- Basic XSS prevention via frontend validation and escaping

Session Handling:

- JWT stored securely (HTTP-only cookie or secure storage on mobile)

Rate Limiting:

- Basic rate limiting (optional if time permits)
- Not mandatory for MVP

No admin endpoints exist in PeerTayo.

3.3 Compatibility Requirements

Web Browsers:

- Google Chrome (latest version)
- Microsoft Edge (latest version)
- Brave (latest version)

Mobile:

- Android API Level 24+ (Android 7.0+)

Screen Responsiveness:

- Mobile: 360px+
- Tablet: 768px+
- Desktop: 1024px+

Operating Systems:

- Windows 10+
- macOS 10.15+
- Linux Ubuntu 20.04+

Deployment Compatibility:

- Backend deployable on Railway/Render/Heroku
- Web deployable on Vercel/Netlify
- Android distributed as APK

3.4 Usability Requirements

User Onboarding:

- A new user should complete:
    - Registration
    - Login
    - Evaluation submission within 5 minutes

Navigation:

- Clear sidebar navigation
- Dashboard as landing page
- Consistent layout across pages

Evaluation Completion:

- Evaluation form should be completable within 3–5 minutes

Error Handling:

- Clear validation messages
- Field-specific error prompts
- Informative success messages

Mobile Usability:

- Touch targets minimum 44x44px
- Scrollable evaluation form
- Clean modal display

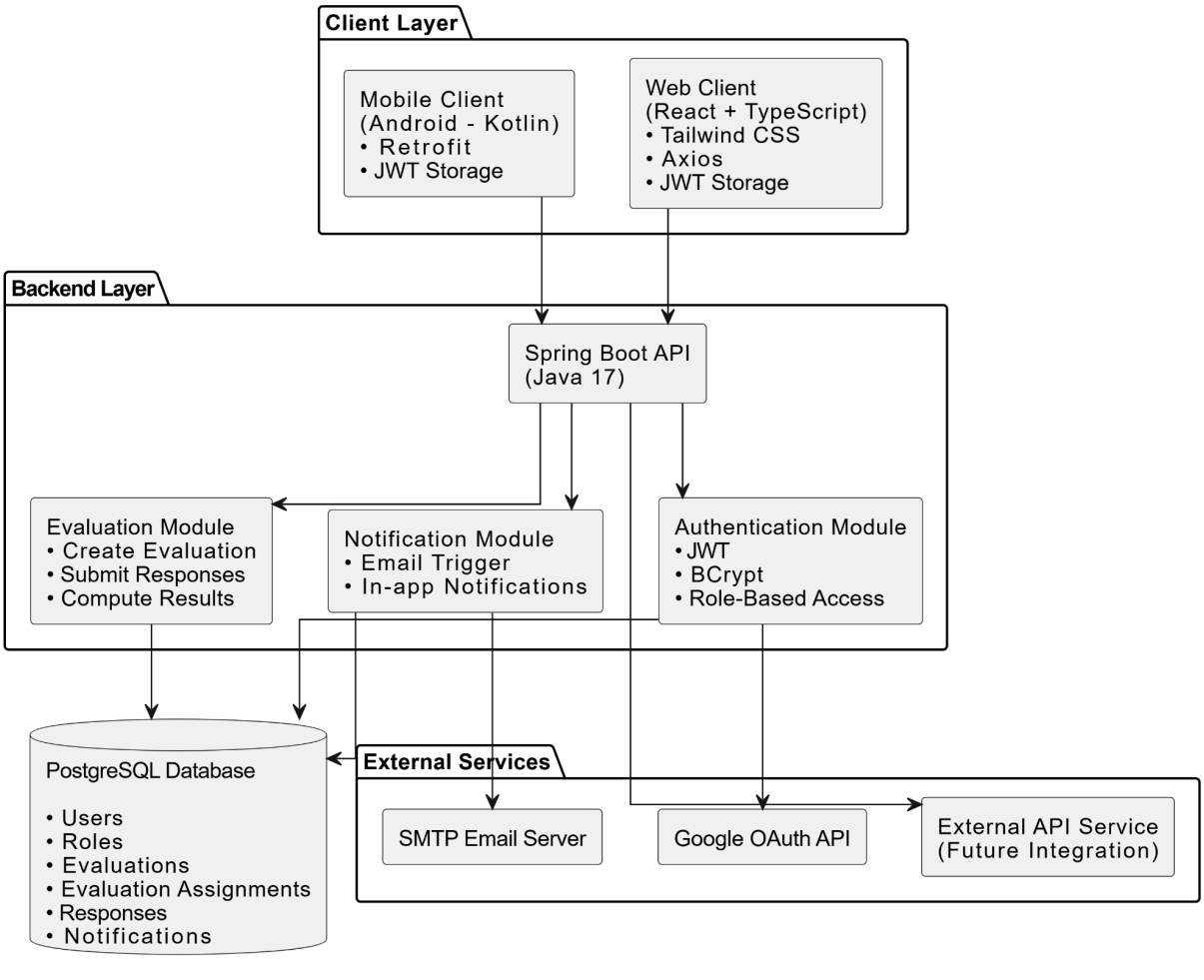
Accessibility:

- Readable font sizes
- Proper color contrast
- Keyboard navigable web interface

4.0 SYSTEM ARCHITECTURE

4.1 Component Diagram

Note: This should be a component diagram



Technology Stack:

- **Backend:**
  Java 17, Spring Boot 3.x, Spring Security, Spring Data JPA, Hibernate, JWT, BCrypt

- **Database:**
  MySQL 8.0+
  MySQL Workbench (Database management & modeling tool)

- **Web Frontend:**
  React 18, TypeScript, Tailwind CSS, Axios

- **Mobile:**
  Kotlin, Jetpack Compose, Retrofit, Room

- **Build Tools:**
  Maven (Backend), npm (Web), Gradle (Android)

- **Deployment:**
  Railway/Render (Backend), Vercel/Netlify (Web), APK Distribution (Mobile)

- **Integrations:**
  Google OAuth 2.0, SMTP Email Service, External REST API

5.0 API CONTRACT & COMMUNICATION

5.1 API Standards

Base URL	https://api.peertayo.com/api/v1
Format	JSON for all requests/responses
Authentication	Bearer token (JWT) in Authorization header
Header Format	Authorization: Bearer {token}
Content Type	application/json

Standard Response Structure	{
	"success": boolean,
	"data": object | null,
	"error": {
	"code": string,
	"message": string,
	"details": object | string | null
	},
	"timestamp": string
	}
	mestamp": string
	}

5.2 Endpoint Specifications

Authentication Endpoints

User Registration

Description	User Registration
API URL	/auth/register
HTTP Request Method	POST
Format	JSON for all requests/responses
Authentication	None

Request Payload	{
	"firstName": "Juan",
	"lastName": "Dela Cruz",
	"email": "juan@email.com",
	"password": "password123"
	}

Response Structure	{
	"success": boolean,
	"data": {
	"user": {
	"email": string,
	"firstName": string,
	"lastName": string
	},
	"accessToken": string,
	"refreshToken": string
	},
	"error": {
	"code": string,
	"message": string,
	"details": object | string | null
	},
	"timestamp": string
	}

User Login

Description	User Login
API URL	/auth/login
HTTP Request Method	POST
Format	JSON for all requests/responses
Authentication	None

Request Payload	{
	"email": "juan@email.com",
	"password": "password123"
	}

Response Structure	{
	"success": true,
	"data": {
	"user": {
	"id": 1,
	"firstName": "Juan",
	"lastName": "Dela Cruz",
	"email": "juan@email.com",
	"roles": ["RESPONDENT"]
	},
	"token": "jwt-token"
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Google OAuth Login

Description	Google OAuth Login
API URL	/auth/google
HTTP Request Method	POST
Format	JSON for all requests/responses
Authentication	None
Request Payload	{ "googleToken": "google-id-token" }
Response Structure	{ "success": true, "data": { "user": { "id": 1, "firstName": "Juan", "lastName": "Dela Cruz", "email": "juan@email.com", "roles": ["RESPONDENT"] }, "token": "jwt-token" }, "error": null, "timestamp": "2026-02-28T10:15:30Z" }

Get Current User

Description	Get currently authenticated user
API URL	/auth/me
HTTP Request Method	GET
Format	JSON for all requests/responses
Authentication	Bearer token (JWT)
Request Payload	None

Response Structure	{
	"success": true,
	"data": {
	"user": {
	"id": 1,
	"firstName": "Juan",
	"lastName": "Dela Cruz",
	"email": "juan@email.com",
	"roles": [
	"RESPONDENT",
	"FACILITATOR"
	]
	}
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

User Logout

Description	Logout current user
API URL	/auth/logout
HTTP Request Method	POST
Format	JSON for all requests/responses
Authentication	Bearer token (JWT)
Request Payload	None

Response Structure	{
	"success": true,
	"data": {
	"message": "Logged out successfully"
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Create Evaluation

Description	Create new evaluation. If user does not yet have FACILITATOR role, system automatically assigns it.
API URL	/evaluations
HTTP Request Method	POST
Format	JSON for all requests/responses
Authentication	Bearer token (JWT) — Role: FACILITATOR

Request Payload	{
	"title": "Peer Evaluation - Group 1",
	"description": "Midterm peer assessment",
	"deadline": "2026-03-10T23:59:00",
	"evaluateeIds": [
	2,
	3,
	4
	],
	"evaluatorIds": [
	5,
	6,
	7
	]
	}

Response Structure	{
	"success": true,
	"data": {
	"evaluation": {
	"id": 10,
	"title": "Peer Evaluation - Group 1",
	"deadline": "2026-03-10T23:59:00",
	"createdBy": 1,
	"status": "ACTIVE"
	}
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Get Pending Evaluations

Description	Returns evaluations assigned to the logged-in user
API URL	/evaluations/pending
HTTP Request Method	GET
Format	JSON for all requests/responses
Authentication	Bearer token (JWT) — Role: RESPONDENT
Request Payload	None
Response Structure	{ "success": true, "data": { "evaluations": [{ "id": 10, "title": "Peer Evaluation - Group 1", "deadline": "2026-03-10T23:59:00", "evaluateeName": "Maria Santos" }] }, "error": null, "timestamp": "2026-02-28T10:15:30Z" }

Submit Evaluation

Description	Submit responses for assigned evaluation
API URL	/evaluations/{id}/submit
HTTP Request Method	POST
Format	JSON for all requests/responses
Authentication	Bearer token (JWT) — Role: RESPONDENT

Request Payload	{
	"responses": [
	{
	"criteriaId": 1,
	"rating": 4
	},
	{
	"criteriaId": 2,
	"rating": 5
	}
	],
	"comment": "Very cooperative member."
	}

Response Structure	{
	"success": true,
	"data": {
	"message": "Evaluation submitted successfully"
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Business Rules:

- Cannot submit after deadline
- Cannot resubmit once completed

Get My Results

Description	Returns aggregated evaluation results for the logged-in user
API URL	/evaluations/my-results
HTTP Request Method	GET
Format	JSON for all requests/responses
Authentication	Bearer token (JWT)
Request Payload	None

Response Structure	{
	"success": true,
	"data": {
	"results": {
	"overallAverage": 4.3,
	"questionAverages": [
	{
	"criteriaId": 1,
	"average": 4.5
	},
	{
	"criteriaId": 2,
	"average": 4.1
	}
	],
	"comments": [
	"Very responsible",
	"Great teamwork"
	]
	}
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Get Created Evaluations

Description	List evaluations created by the logged-in user
API URL	/evaluations/created
HTTP Request Method	GET
Format	JSON for all requests/responses
Authentication	Bearer token (JWT) — Role: FACILITATOR
Request Payload	None

Response Structure	{
	"success": true,
	"data": {
	"evaluations": [
	{
	"id": 10,
	"title": "Peer Evaluation - Group 1",
	"deadline": "2026-03-10T23:59:00",
	"submissionProgress": "5/7"
	}
	]
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Get Evaluation Results (Facilitator)

Description	View detailed aggregated results of a specific evaluation
API URL	/evaluations/{id}/results
HTTP Request Method	GET
Format	JSON for all requests/responses
Authentication	Bearer token (JWT) — Role: FACILITATOR
Request Payload	None

Response Structure	{
	"success": true,
	"data": {
	"evaluationId": 10,
	"evaluatees": [
	{
	"userId": 2,
	"overallAverage": 4.2,
	"criteriaAverages": [...]
	}
	]
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Get Evaluation Results (Facilitator)

Description	View detailed aggregated results of a specific evaluation
API URL	/evaluations/{id}/results
HTTP Request Method	GET
Format	JSON for all requests/responses
Authentication	Bearer token (JWT) — Role: FACILITATOR
Request Payload	None

Response Structure	{
	"success": true,
	"data": {
	"evaluationId": 10,
	"evaluatees": [
	{
	"userId": 2,
	"overallAverage": 4.2,
	"criteriaAverages": [...]
	}
	]
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Get Notifications

Description	Retrieve all notifications for the logged-in user
API URL	/notifications
HTTP Request Method	GET
Format	JSON for all requests/responses
Authentication	Bearer token (JWT)
Request Payload	None

Response Structure	{
	"success": true,
	"data": {
	"notifications": [
	{
	"id": 1,
	"message": "You have a new evaluation assigned.",
	"isRead": false,
	"createdAt": "2026-02-25T10:30:00"
	}
	]
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

Mark Notification as Read

Description	Mark a specific notification as read
API URL	/notifications/{id}/read
HTTP Request Method	PUT
Format	JSON for all requests/responses
Authentication	Bearer token (JWT)
Request Payload	None

Response Structure	{
	"success": true,
	"data": {
	"message": "Notification marked as read"
	},
	"error": null,
	"timestamp": "2026-02-28T10:15:30Z"
	}

5.3 Error Handling

HTTP Status Codes

200 OK – Successful request
201 Created – Resource created
400 Bad Request – Invalid input
401 Unauthorized – Authentication failed
403 Forbidden – Insufficient permissions
404 Not Found – Resource not found
409 Conflict – Duplicate entry
422 Unprocessable Entity – Business rule violation
500 Internal Server Error – Server error

Error Code Examples

AUTH-001
Invalid
Credentials
	{
	"success": false,
	"data": null,
	"error": {
	"code": "AUTH-001",
	"message": "Invalid credentials",
	"details": "Email or password is incorrect"
	},
	"timestamp": "2026-02-28T10:15:30Z"
	}

AUTH-002
Unauthorized Access
	{
	"success": false,
	"data": null,
	"error": {
	"code": "AUTH-002",
	"message": "Unauthorized",
	"details": "Bearer token is missing or expired"
	},
	"timestamp": "2026-02-28T10:15:30Z"
	}

AUTH-003
Forbidden Role
	{
	"success": false,
	"data": null,
	"error": {
	"code": "AUTH-003",
	"message": "Forbidden",
	"details": "You do not have the required role to access this resource"
	},
	"timestamp": "2026-02-28T10:15:30Z"
	}

VALID-001
Validation Failed
	{
	"success": false,
	"data": null,
	"error": {
	"code": "VALID-001",
	"message": "Validation failed",
	"details": {
	"email": "Email is required",
	"password": "Must be at least 8 characters"
	}
	},
	"timestamp": "2026-02-28T10:15:30Z"
	}

EVAL-001
Deadline Passed
	{
	"success": false,
	"data": null,
	"error": {
	"code": "EVAL-001",
	"message": "Submission not allowed",
	"details": "Cannot submit evaluation after the deadline"
	},
	"timestamp": "2026-02-28T10:15:30Z"
	}

EVAL-002
Already Submitted
	{
	"success": false,
	"data": null,
	"error": {
	"code": "EVAL-002",
	"message": "Resubmission not allowed",
	"details": "You have already submitted this evaluation"
	},
	"timestamp": "2026-02-28T10:15:30Z"
	}

NOTIF-001
Notification Not Found
	{
	"success": false,
	"data": null,
	"error": {
	"code": "NOTIF-001",
	"message": "Notification not found",
	"details": "No notification exists with the given ID"
	},
	"timestamp": "2026-02-28T10:15:30Z"
	}

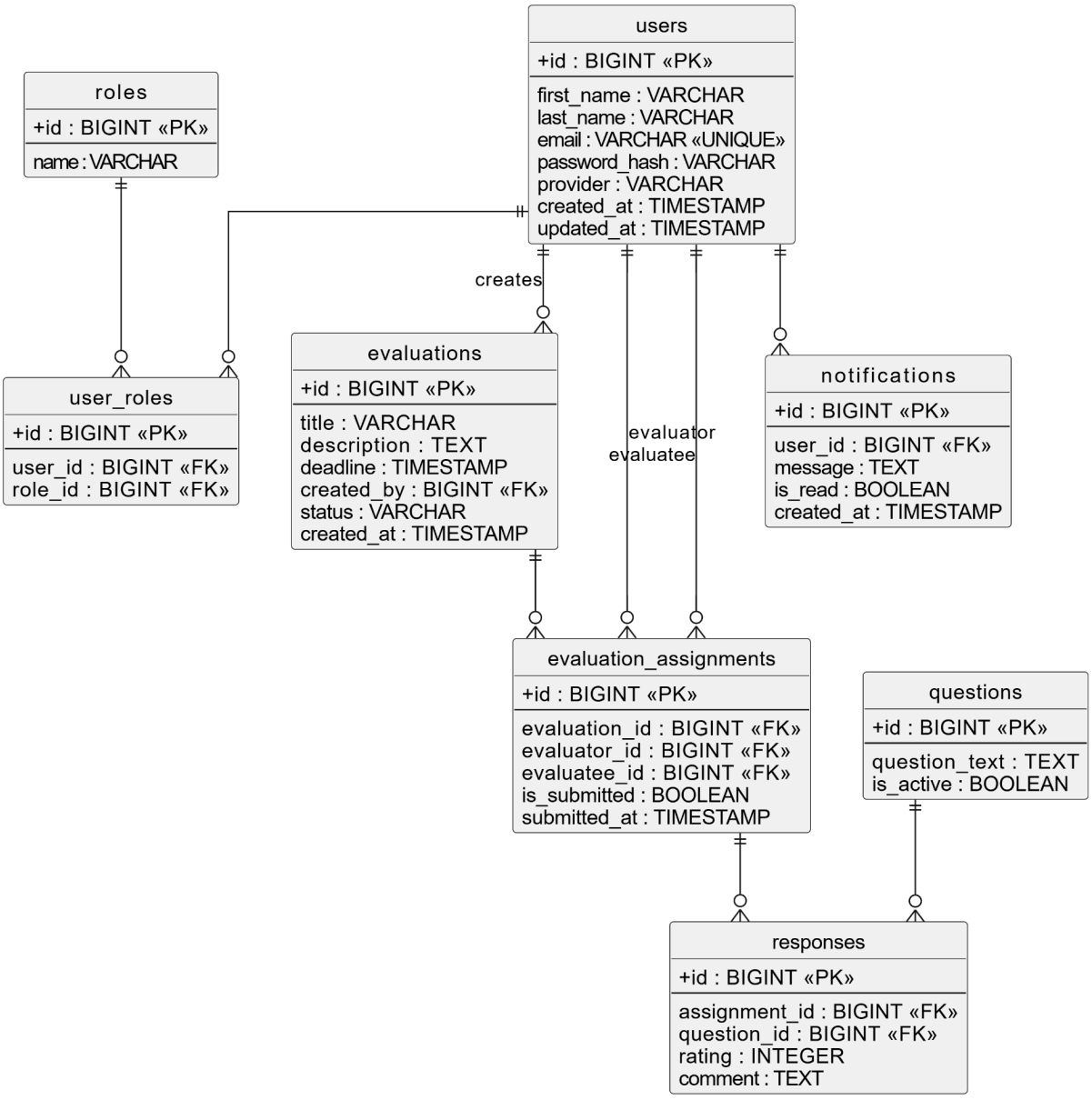
SRV-001
Internal Server Error
	{
	"success": false,
	"data": null,
	"error": {
	"code": "SRV-001",
	"message": "Internal server error",
	"details": "An unexpected error occurred. Please try again later."
	},
	"timestamp": "2026-02-28T10:15:30Z"
	}

Common Error Codes

AUTH-001 – Invalid credentials
AUTH-002 – Token expired
AUTH-003 – Insufficient permissions
VALID-001 – Validation failed
DB-001 – Resource not found
DB-002 – Duplicate entry
BUSINESS-001 – Deadline passed
BUSINESS-002 – Evaluation already submitted
SYSTEM-001 – Internal server error

6.0 DATABASE DESIGN

6.1 Entity Relationship Diagram



Detailed Relationships

Many-to-Many:
User ↔ Role

- A user can have multiple roles (RESPONDENT, FACILITATOR)
- A role can belong to multiple users
- Implemented via USER_ROLES table

One-to-Many:
User → Evaluations

- A facilitator can create multiple evaluations

One-to-Many:
Evaluation → EvaluationAssignments

- An evaluation contains multiple evaluator–evaluatee assignments

One-to-Many:
User → EvaluationAssignments

- A user can be assigned to evaluate multiple users

One-to-Many:
EvaluationAssignment → Responses

- Each assignment contains multiple criteria responses

One-to-Many:
User → Notifications

- A user can receive multiple notifications

One-to-Many:
Question → Responses

- Each question can have multiple response records

Key Tables

users – Stores user account information
roles – Stores available system roles
user_roles – Junction table for many-to-many role relationship
evaluations – Stores evaluation records created by facilitators
evaluation_assignments – Links evaluators to evaluatees
questions – Stores predefined evaluation questions
responses – Stores rating answers per question
notifications – Stores user notifications

Table Structure Summary

users

- id (PK)
- first_name
- last_name
- email (unique)
- password_hash
- provider (LOCAL / GOOGLE)
- created_at
- updated_at

roles

- id (PK)
- name (RESPONDENT / FACILITATOR)

user_roles

- id (PK)
- user_id (FK → users.id)
- role_id (FK → roles.id)

evaluations

- id (PK)
- title
- description
- deadline
- created_by (FK → users.id)
- status (ACTIVE / CLOSED)
- created_at

evaluation_assignments

- id (PK)
- evaluation_id (FK → evaluations.id)
- evaluator_id (FK → users.id)
- evaluatee_id (FK → users.id)
- is_submitted (boolean)

- submitted_at

criteria

- id (PK)
- title (e.g., Communication Skills, Leadership, Teamwork)
- description (main explanation of the criterion)
- is_active (boolean)

(Contains the predefined 10 evaluation criteria.)

rating

- id (PK)
- assignment_id (FK → evaluation_assignments.id)
- criterion_id (FK → criteria.id)
- rating (1–5)
- is_active (boolean)

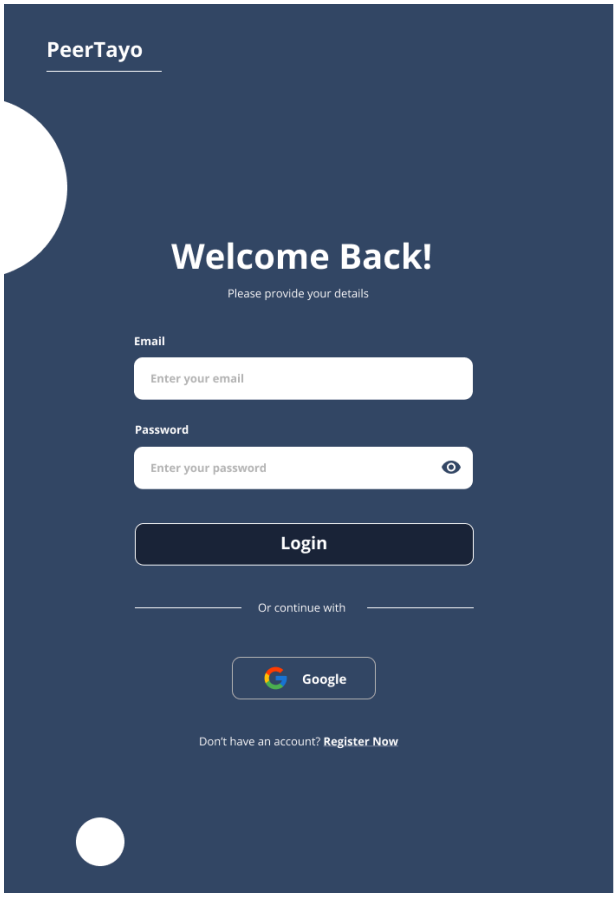
notifications

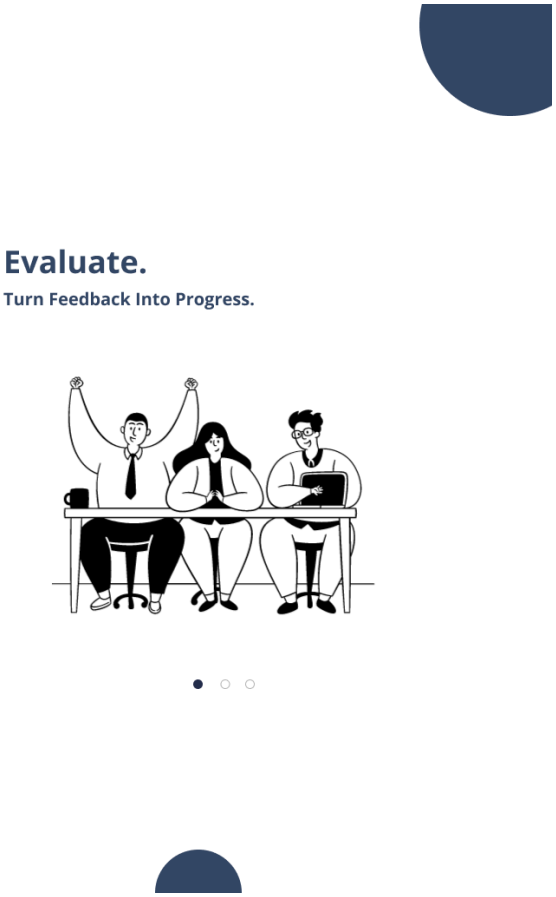
- id (PK)
- user_id (FK → users.id)
- message
- is_read (boolean)
- created_at

7.0 UI/UX DESIGN

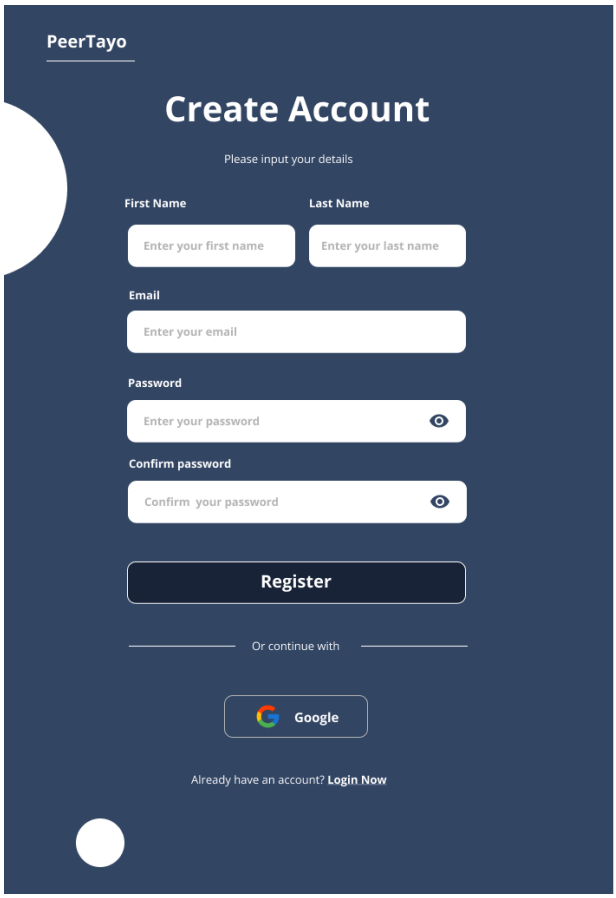
7.1 Web Application Wireframes

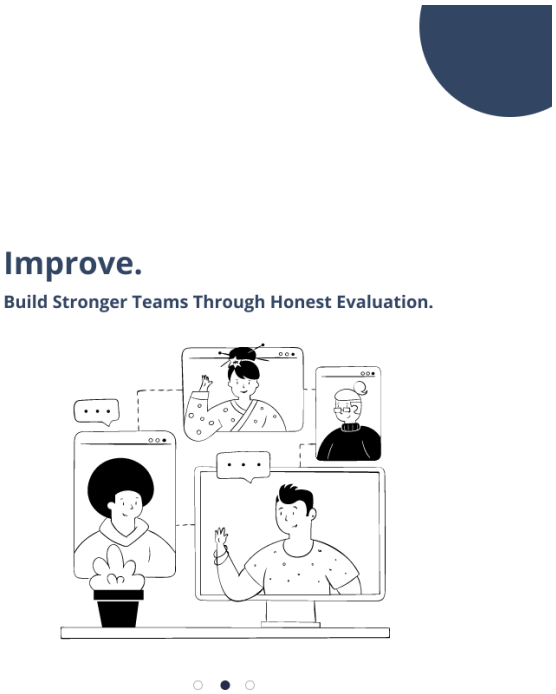
Login



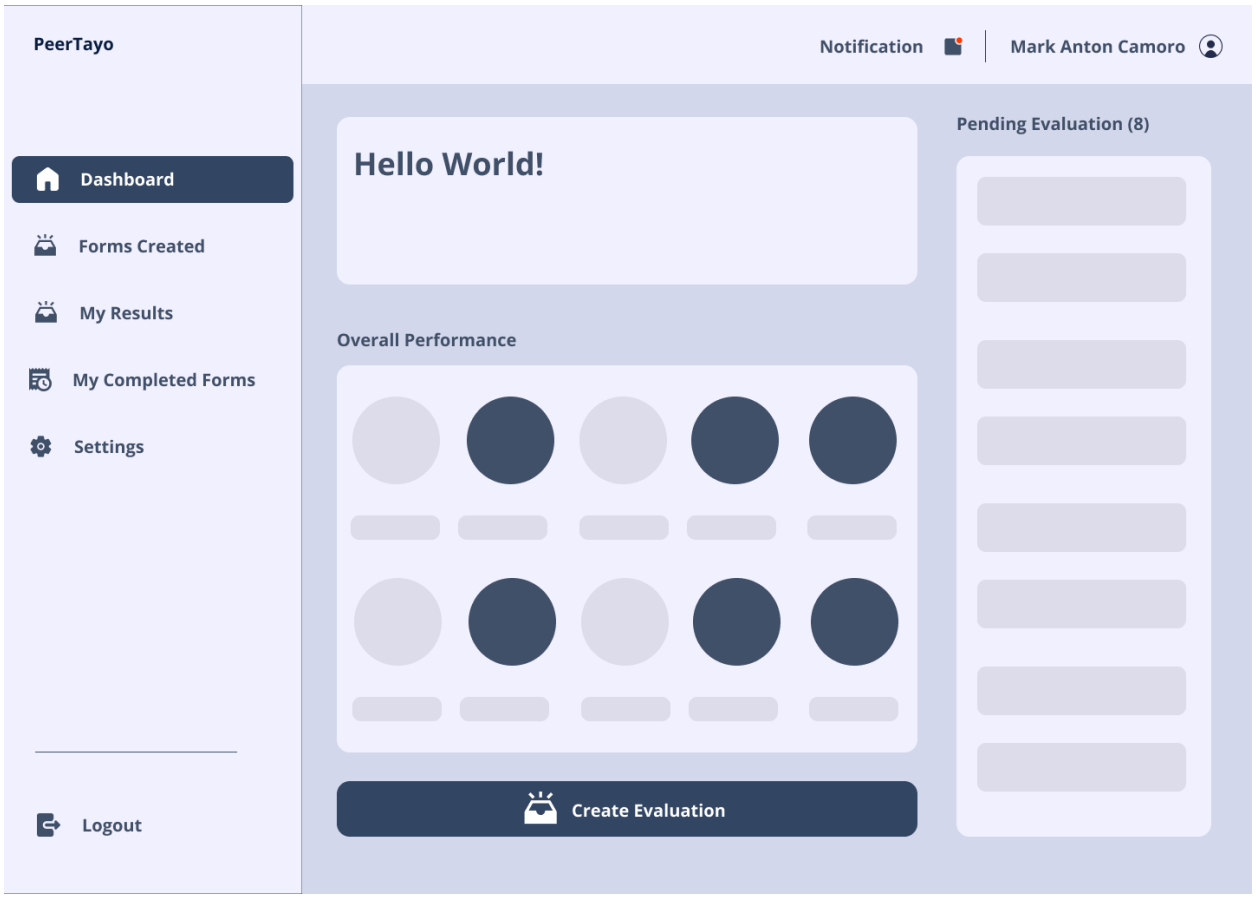


Register


PeerTayo
Create Account
Please input your details
First Name
Last Name
Enter your first name
Enter your last name
Email
Enter your email
Password
Enter your password
Confirm password
Confirm your password
Register
Or continue with
Google
Already have an account? Login Now

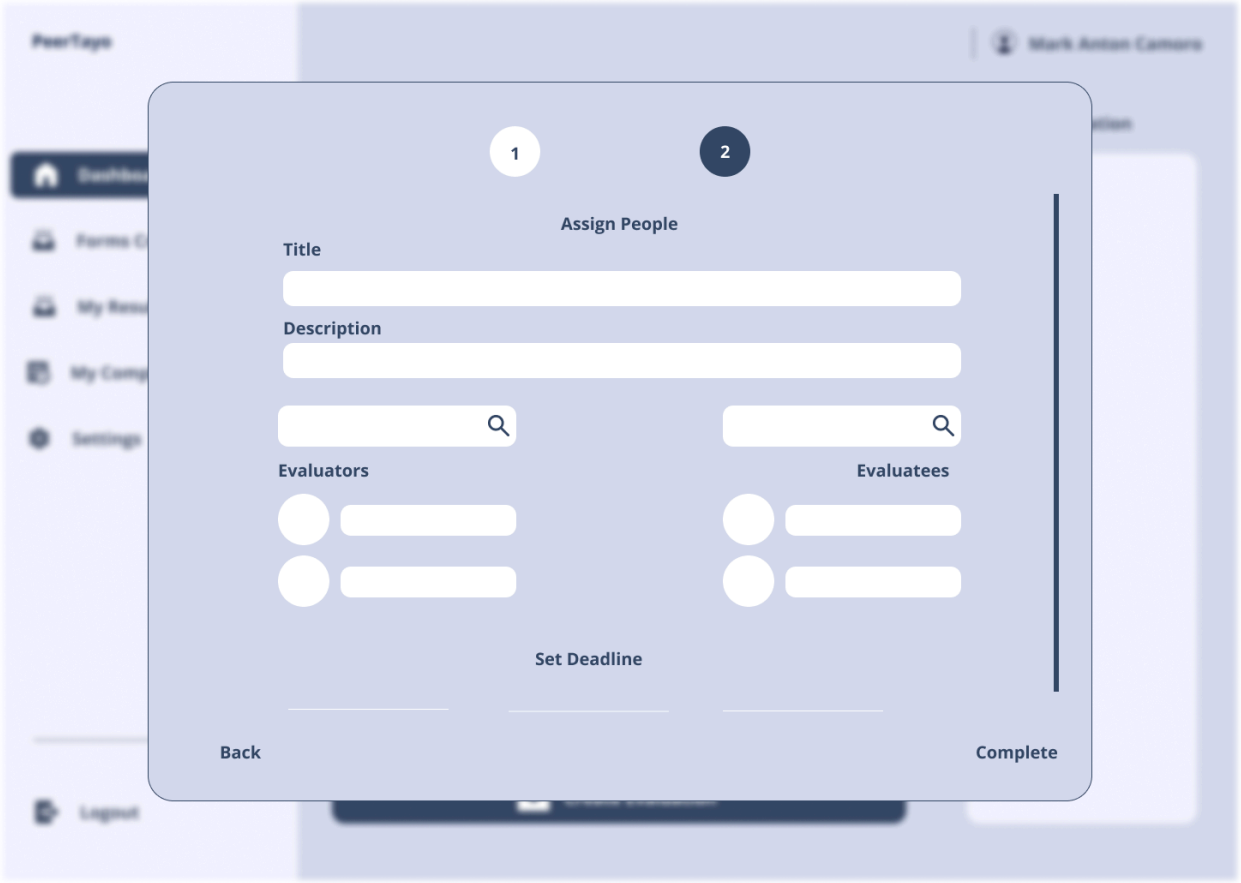

Improve.
Build Stronger Teams Through Honest Evaluation.

Dashboard

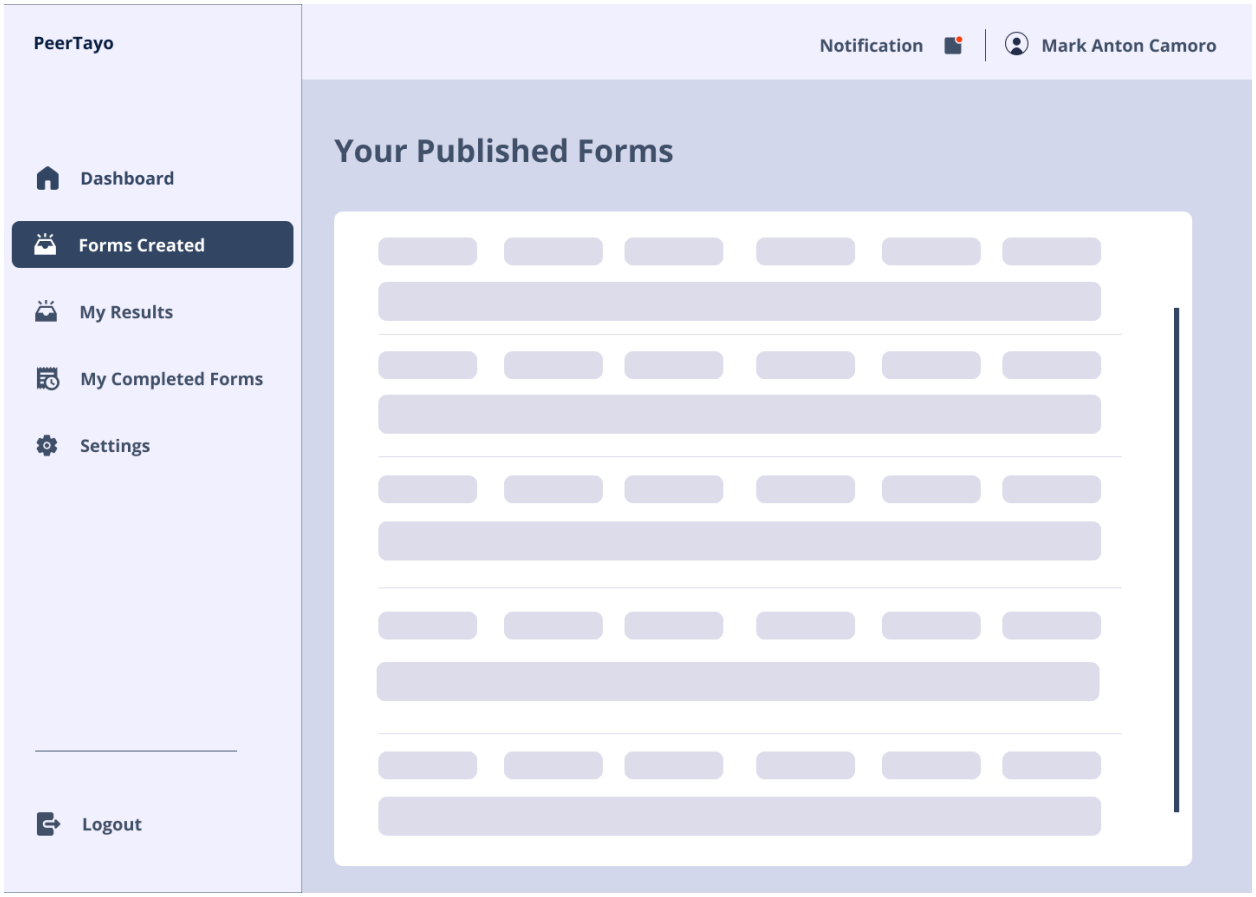

PeerTayo
Notification
Mark Anton Camoro
Dashboard
Forms Created
My Results
My Completed Forms
Settings
Logout
Hello World!
Overall Performance
Create Evaluation
Pending Evaluation (8)

Create Forms 1

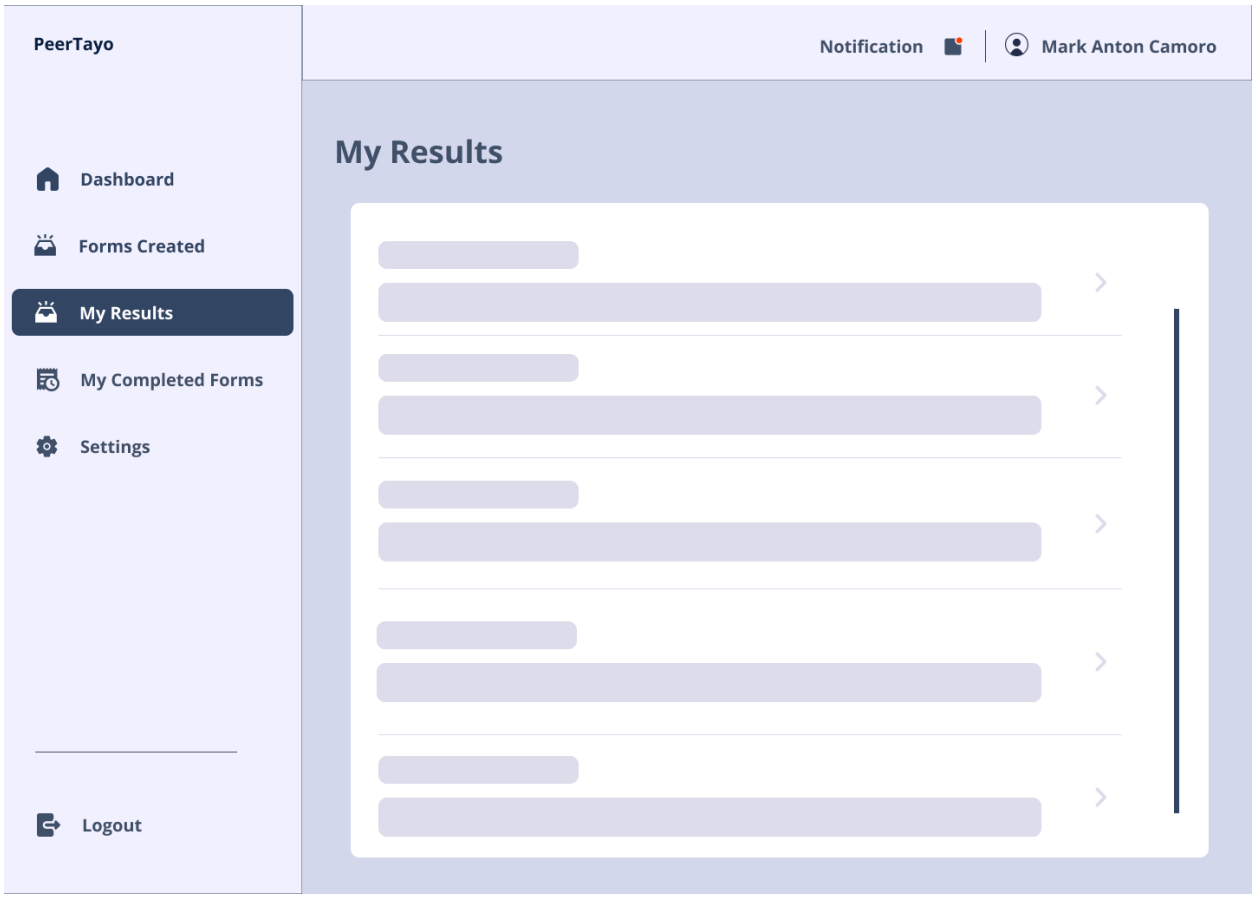
Create Forms 2



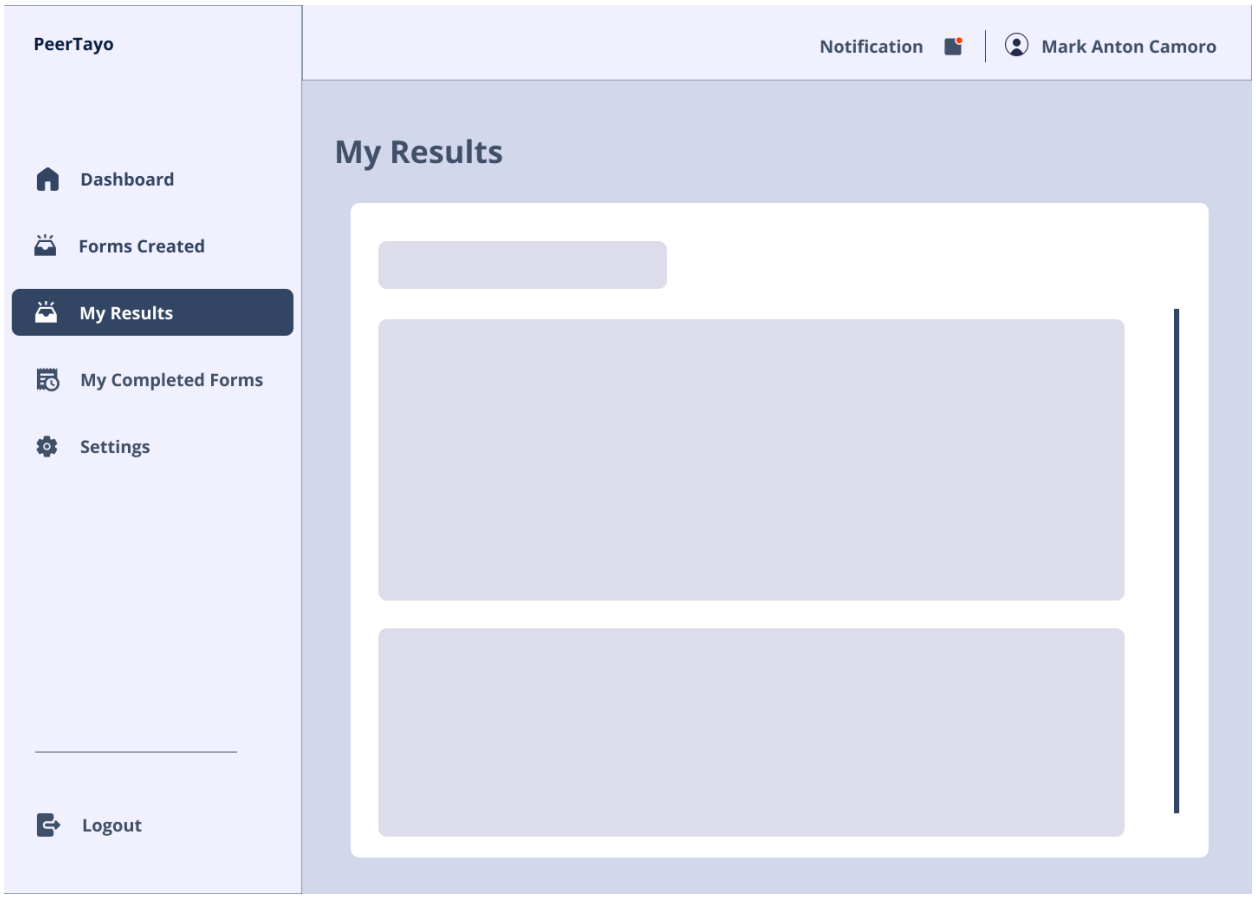
Forms Created



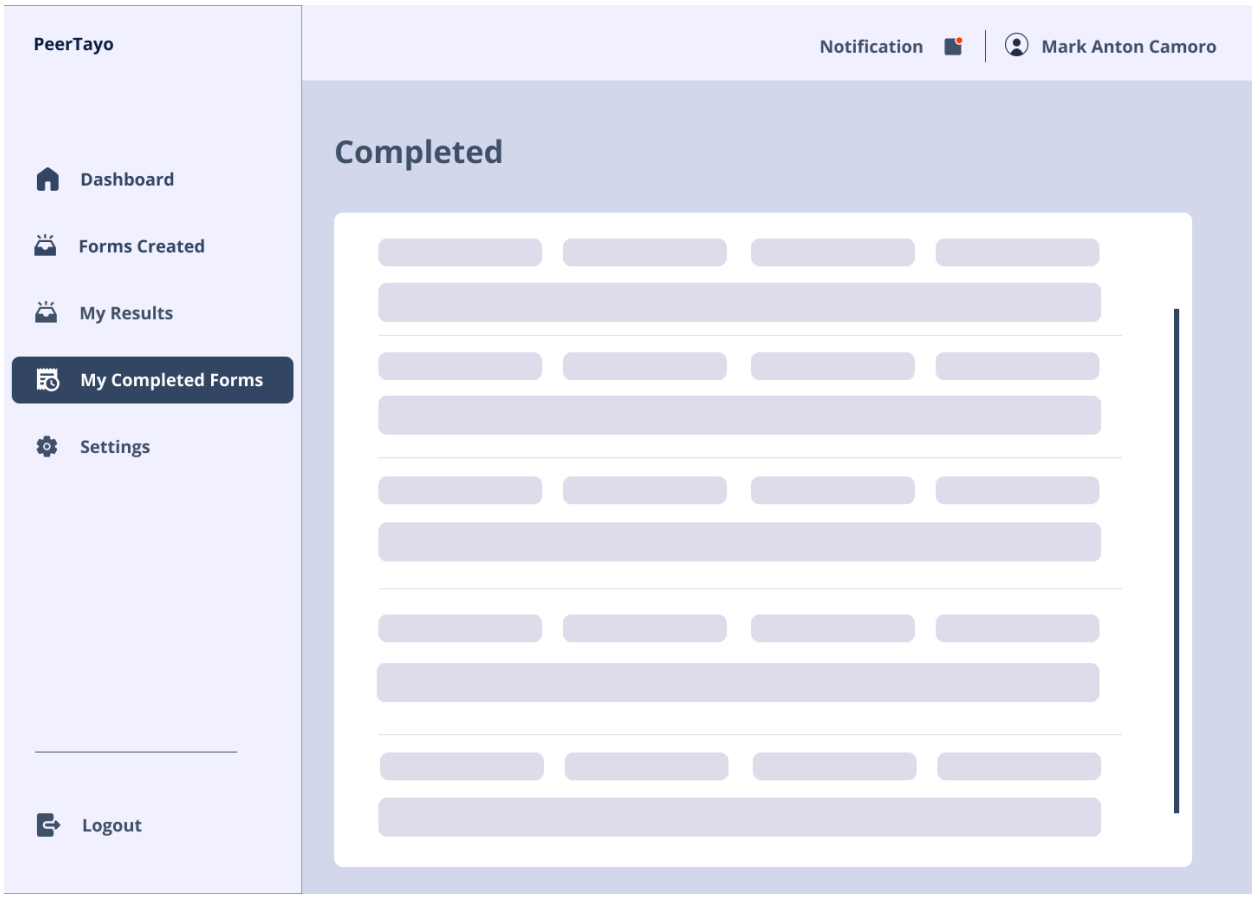
My Results


PeerTayo
Notification
Mark Anton Camoro
Dashboard
Forms Created
My Results
My Completed Forms
Settings
Logout
My Results

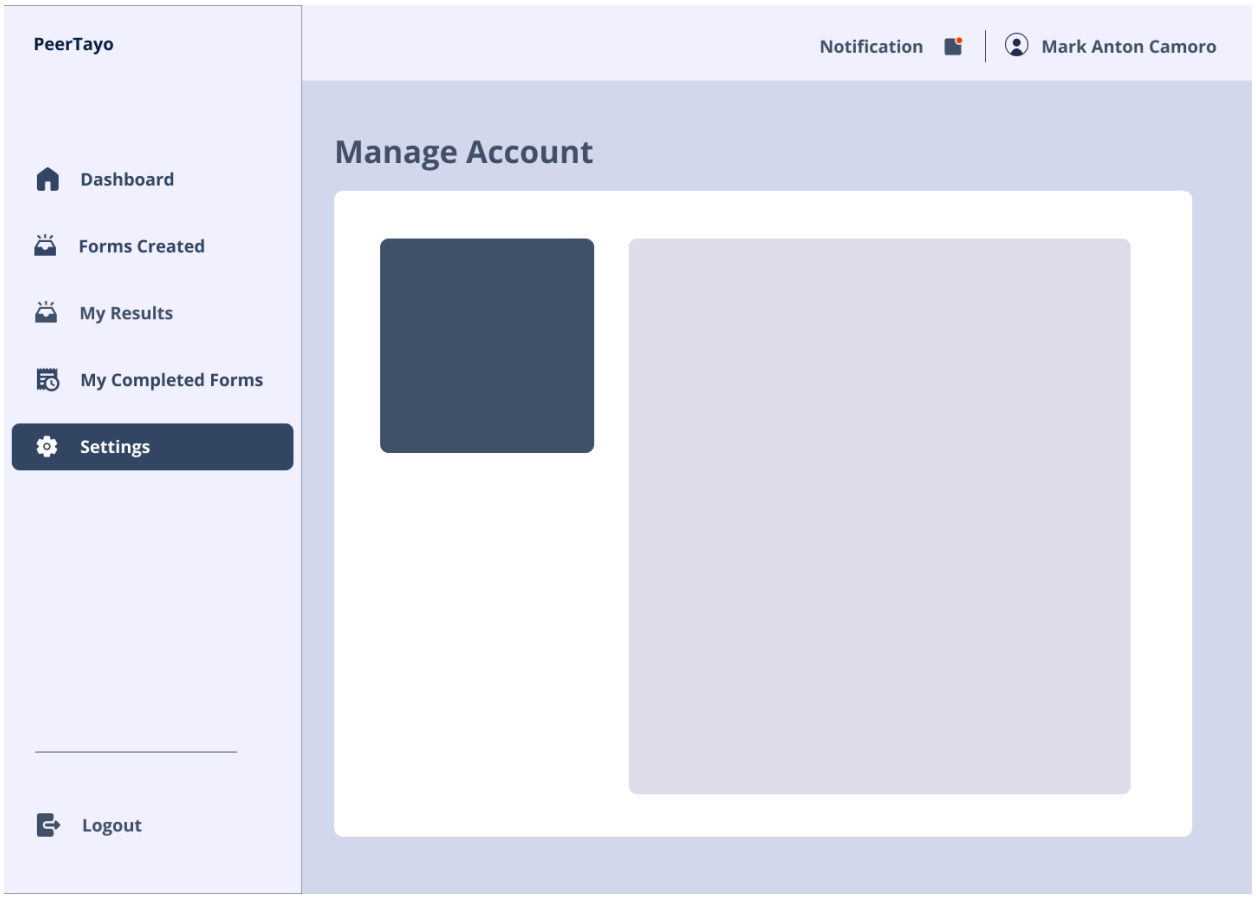
View My Results



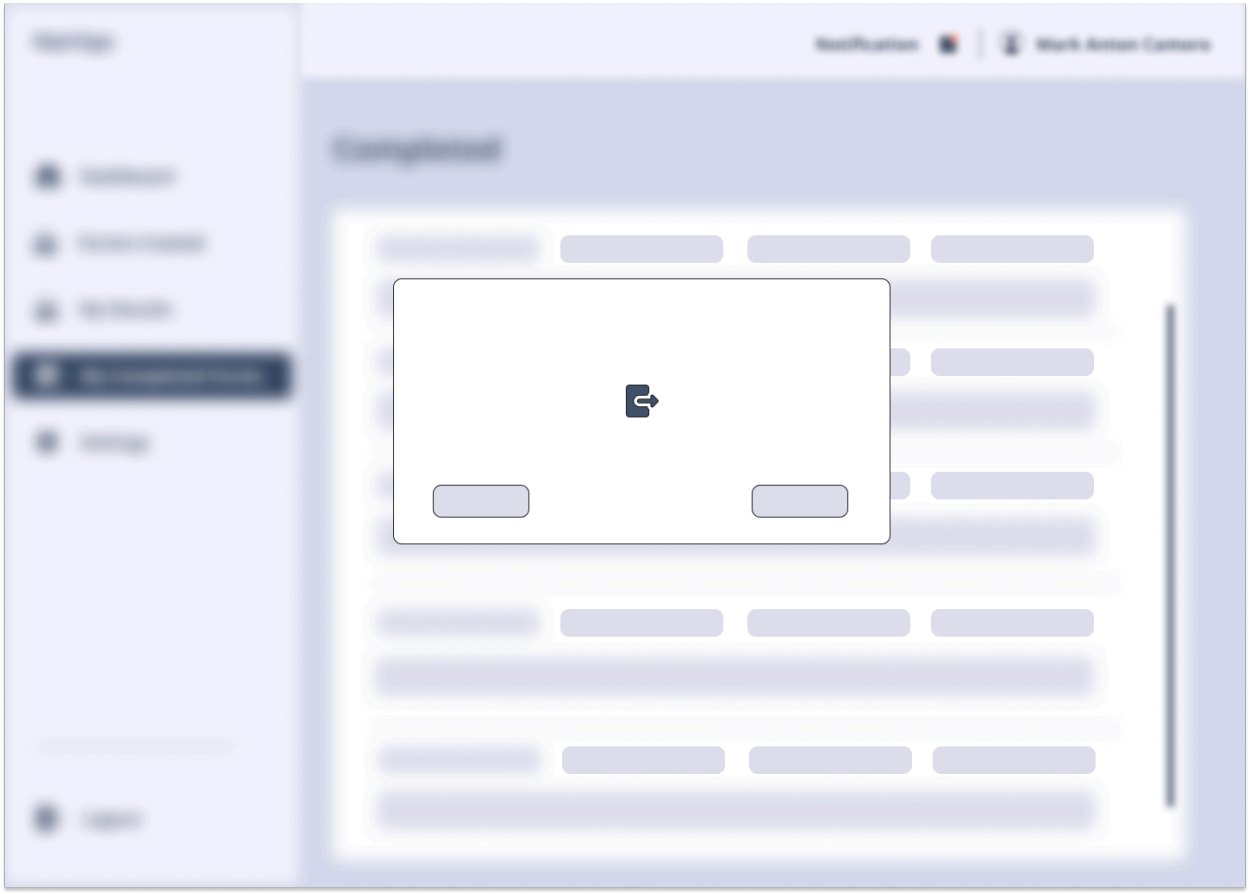
My Completed Forms



Settings

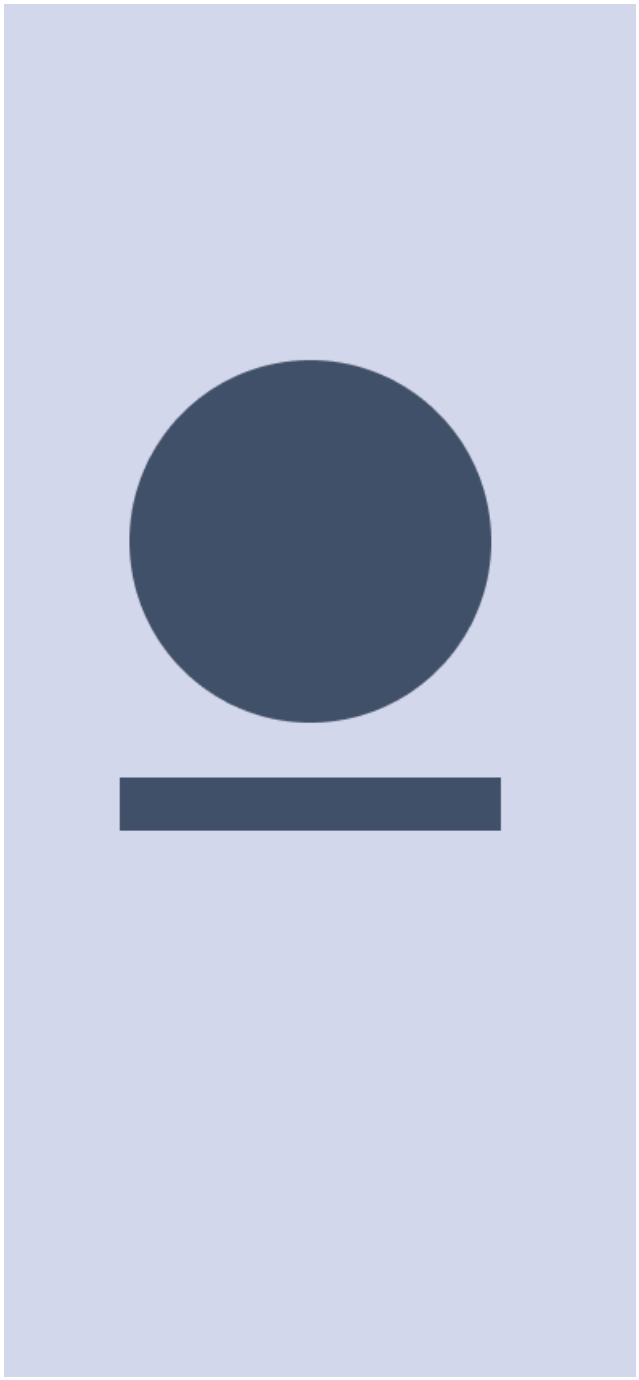

PeerTayo
Notification
Mark Anton Camoro
Dashboard
Forms Created
My Results
My Completed Forms
Settings
Logout
Manage Account

Logout

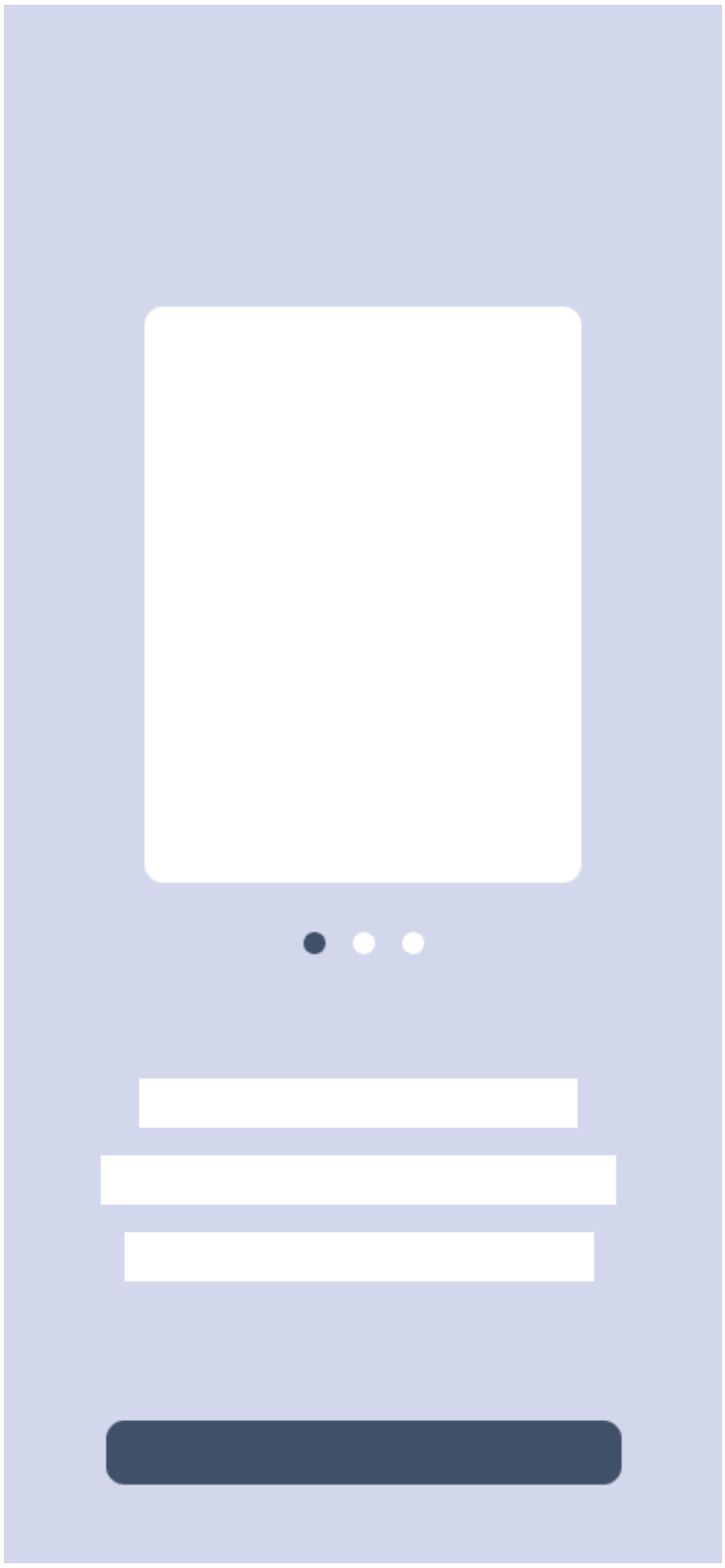


7.2 Mobile Application Wireframes

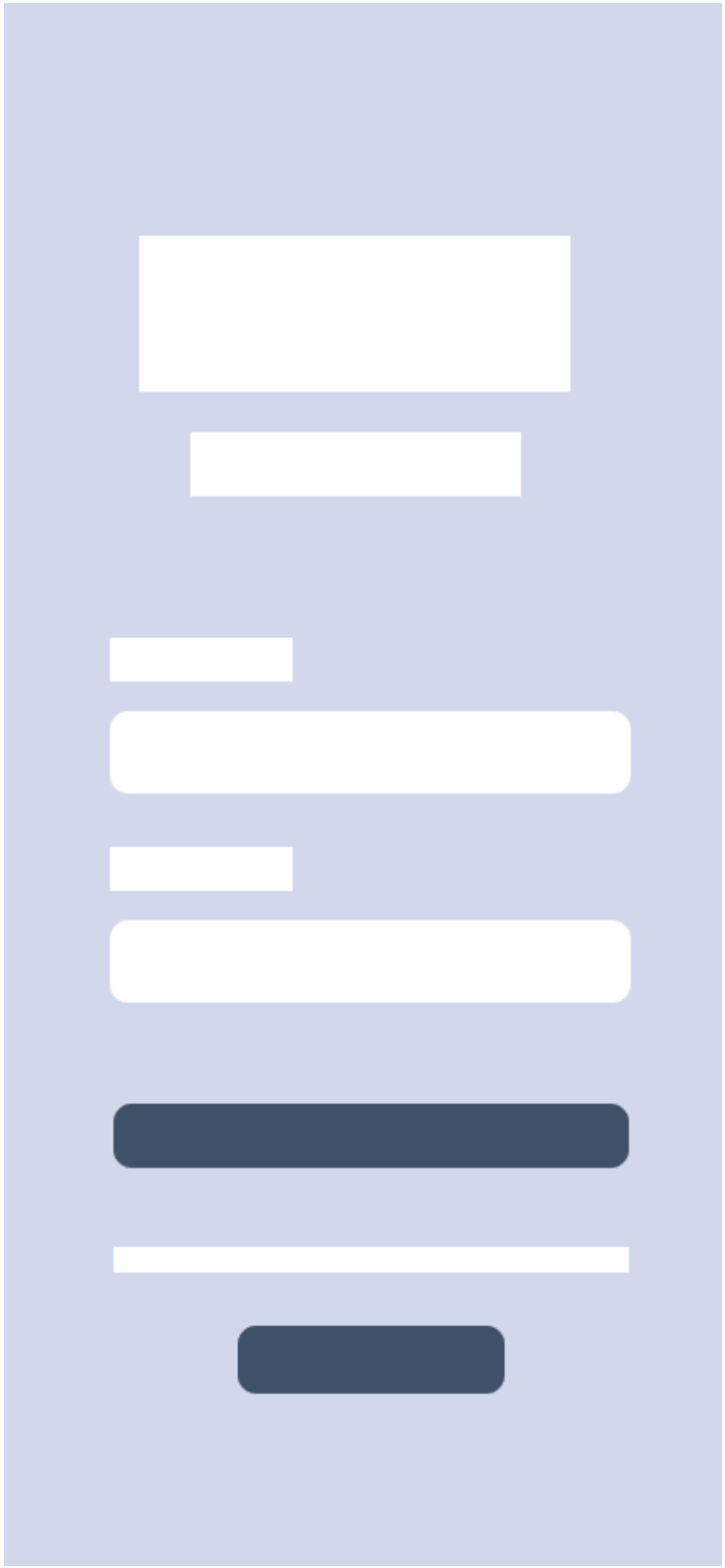
Splash Screen



Get Started



Login



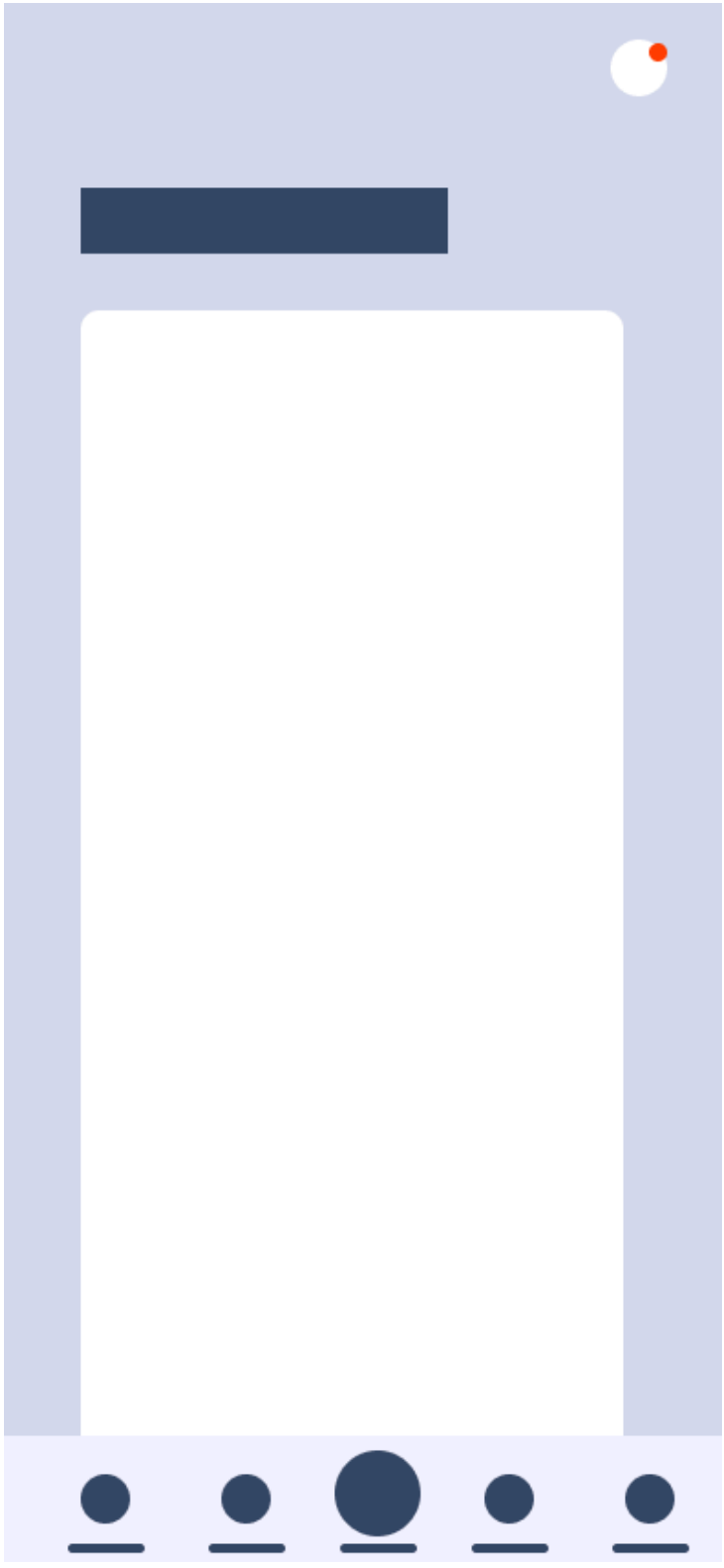
Register

Dashboard

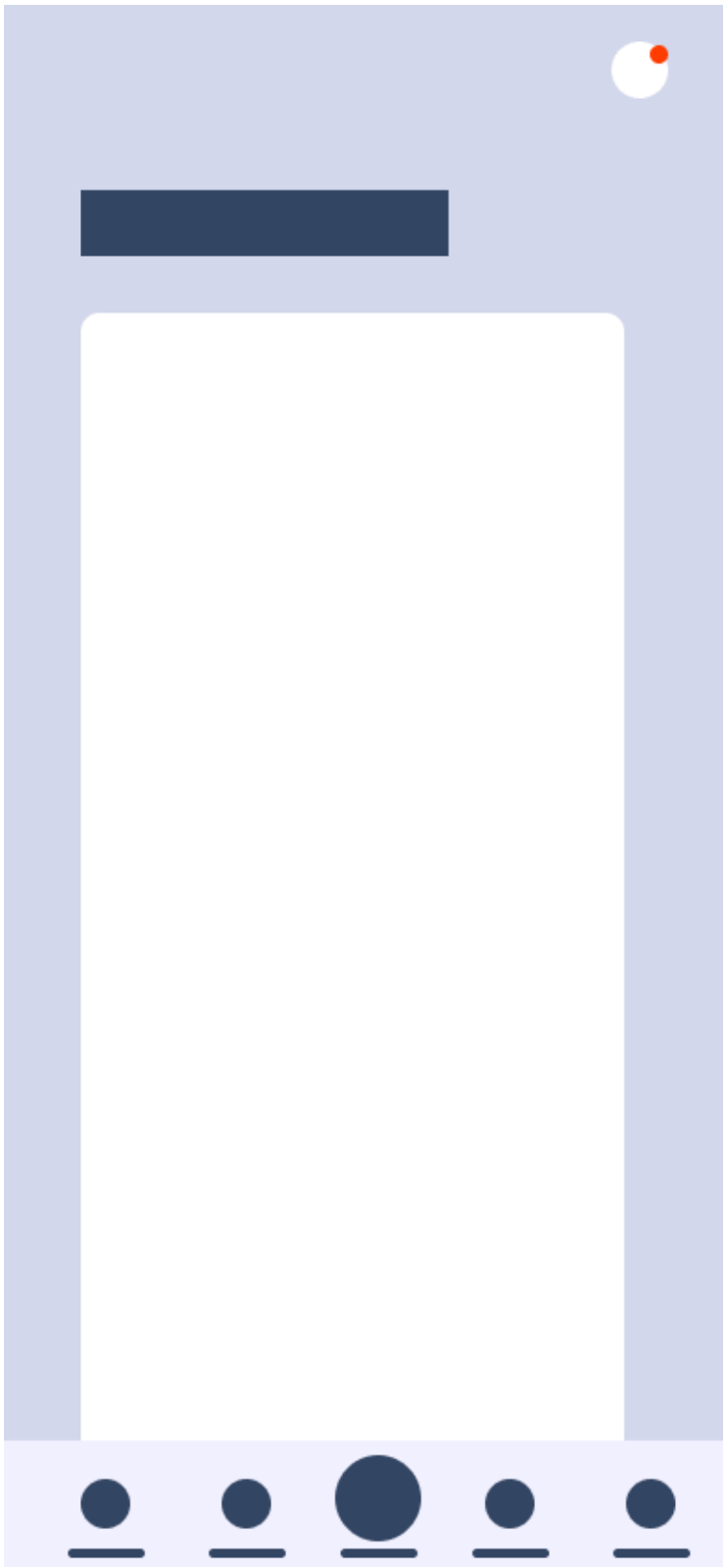
Create Forms 1

Create Forms 2

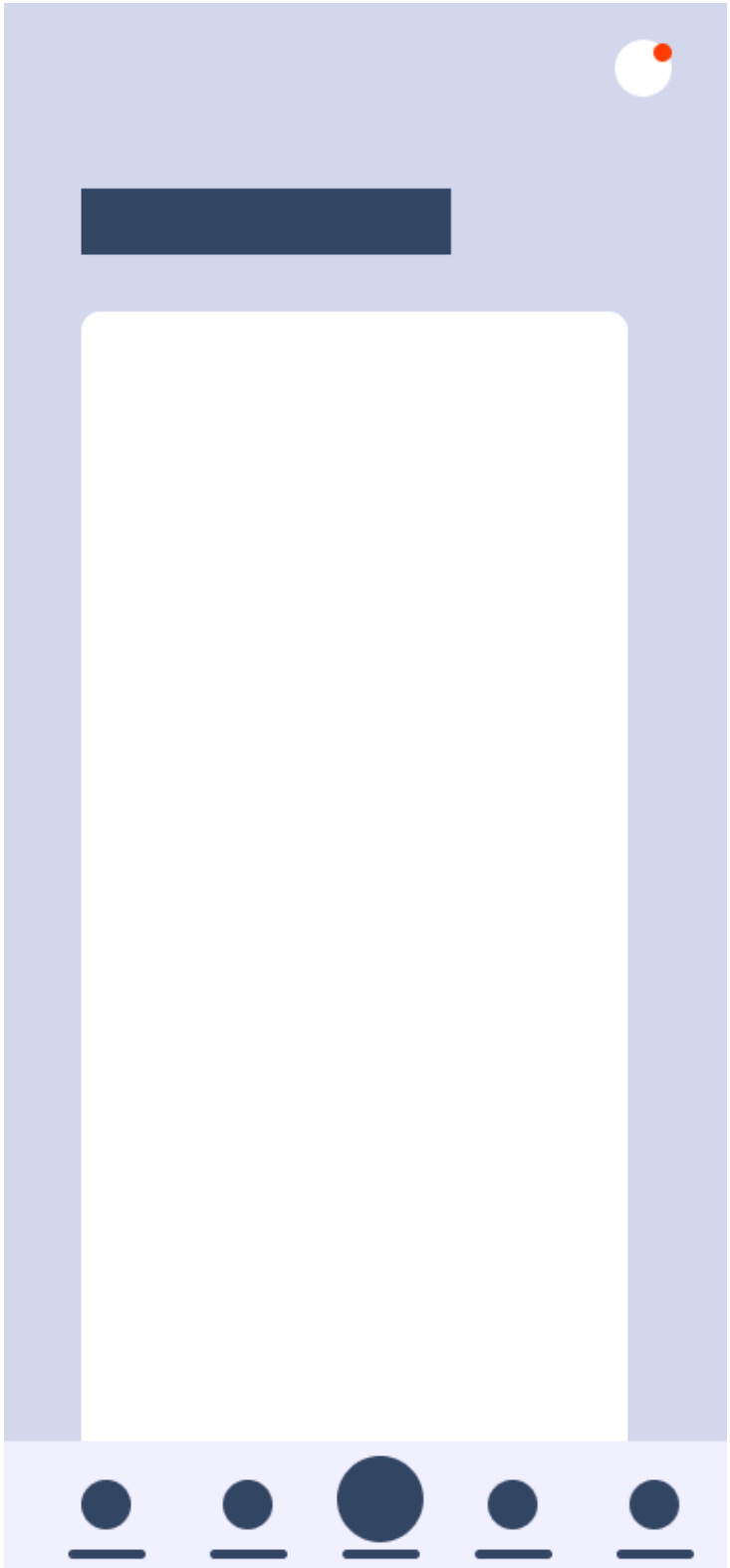
Forms Created



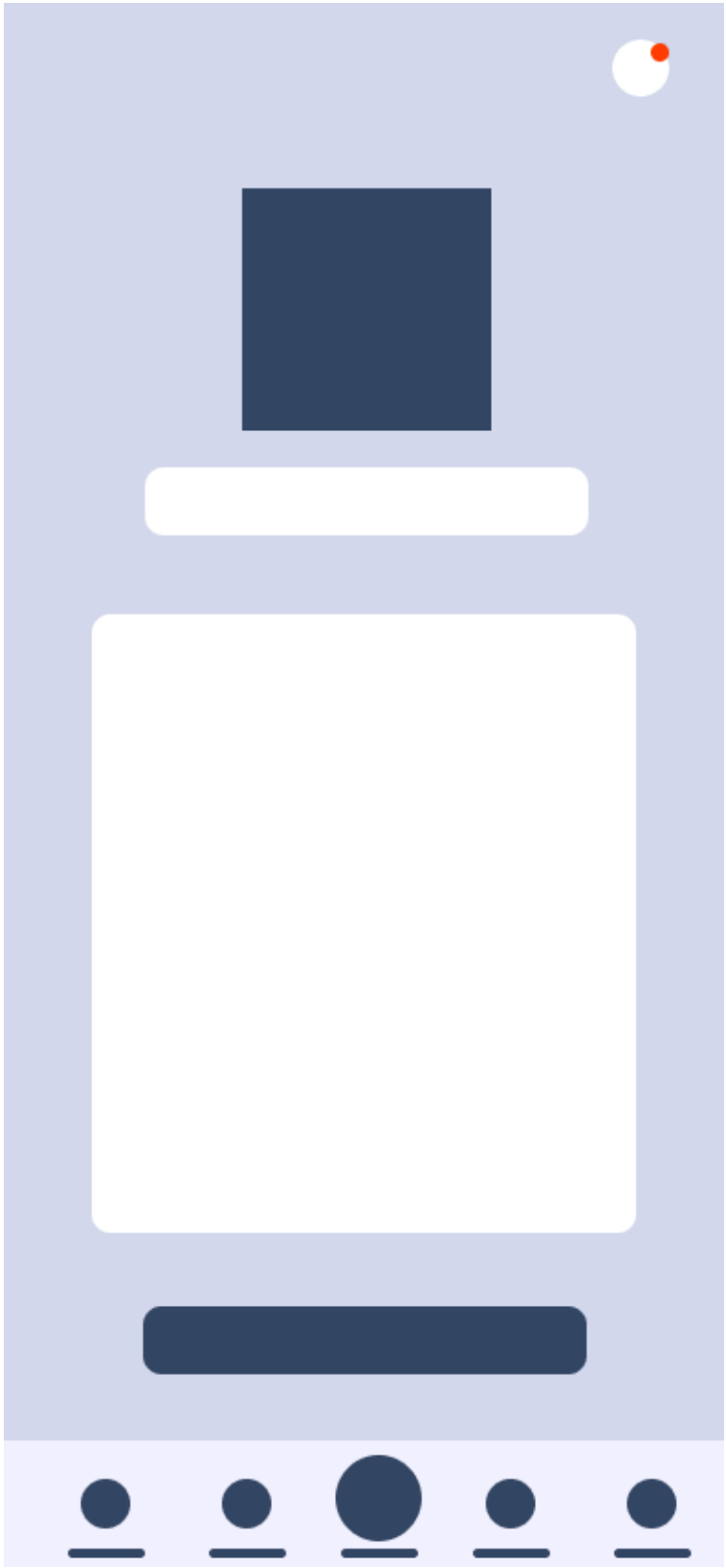
My Results



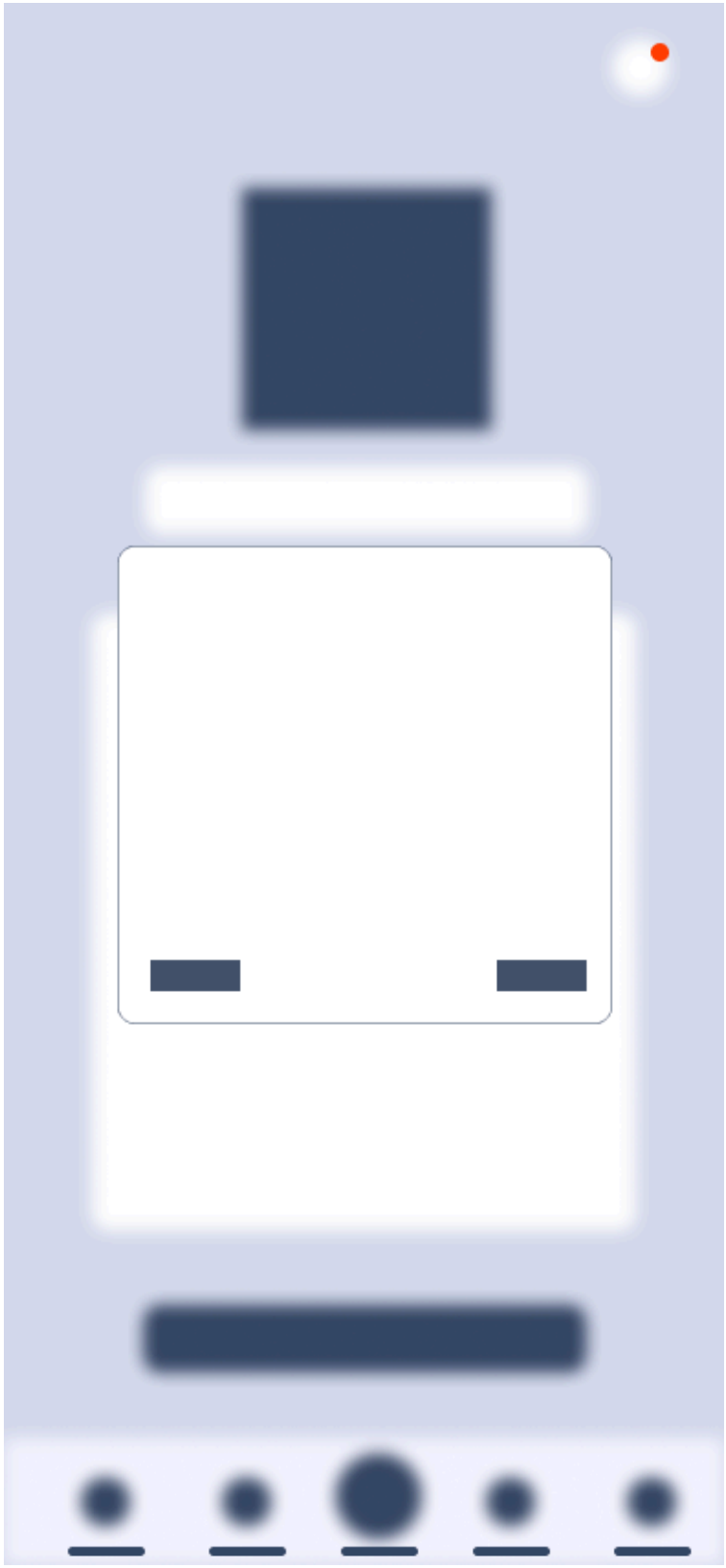
My Completed Forms



Settings



Logout



Design System:

- **Colors:** Primary (#3F5373), Secondary Background (#F4F6FA), Accent (#2F3E57), Text Primary (Dark Blue-Gray), Cards (#FFFFFF)
- **Typography:** Inter font family, H1–H3 for headings, 16px body, 14px captions
- **Spacing:** Base 16px, multiples for padding/margin
- **Components:** Rounded buttons & inputs (12px), cards (12px), badges, minimal icons
- **Responsive:** Mobile-first 390x844 layout, scrollable lists, bottom navigation

8.0 PLAN

8.1 Project Timeline

Phase 1: Planning & Design (Week 1–2)

Week 1: Requirements & Architecture

Day 1–2: Project setup and documentation

- Finalize PeerTayo system scope
- Define user roles (Student, Instructor, Admin)
- Set up project repository structure (Backend, Web, Mobile)

Day 3–4: Complete FRS and NFR

- Functional Requirements (Form creation, Evaluation submission, Results viewing, Notifications, Settings)
- Non-Functional Requirements (Security, Performance, Scalability, Usability)

Day 5–7: System architecture design

- Define overall system architecture (Web + Mobile + Backend API)

- Define authentication flow (Google OAuth + JWT)
- Define high-level database schema
- Define integration between services

Week 2: Detailed Design

Day 1–2: API specification

- Authentication endpoints (Google OAuth, JWT)
- User profile endpoints
- Evaluation form CRUD endpoints
- Submission endpoints
- Results aggregation endpoints
- Notification endpoints

Day 3–4: Database design

- User entity (firstName, lastName, email, role)
- EvaluationForm entity
- Question entity
- EvaluationAssignment entity
- Submission entity
- Response entity
- Notification entity

Day 5–6: UI/UX wireframes

- Login screen (Google OAuth + email login)
- Dashboard
- Forms Created
- My Results
- My Completed Forms
- Settings
- Mobile UI layout in Jetpack Compose

Day 7: Implementation plan finalization

- Sprint task breakdown
- API task assignment
- Frontend screen mapping
- Testing strategy planning

Phase 2: Backend Development (Week 3–4)

Week 3: Foundation

Day 1: Spring Boot setup with dependencies

- Spring Security
- Spring Data JPA
- MySQL configuration

- JWT configuration

Day 2: Database configuration and entities

- Create all entities and relationships
- Configure repositories
- Implement migration scripts

Day 3: Authentication implementation

- Google OAuth integration
- JWT token generation & validation
- Role-based access control

Day 4: User management endpoints

- Register (first name, last name, email)
- Profile update
- Role assignment
- User listing (Admin)

Day 5: Evaluation Form CRUD operations

- Create form
- Add questions (rating, text, scale)
- Edit form
- Delete form

- Publish form

Week 4: Core Features

Day 1: Evaluation assignment functionality

- Assign forms to specific students
- Set deadlines
- Send notifications

Day 2: Submission functionality

- Submit evaluation
- Validate required answers
- Prevent duplicate submission

Day 3: Results aggregation logic

- Compute average ratings
- Anonymous data handling
- Generate summary statistics

Day 4: Error handling and validation

- Form validation
- Security validation
- Input sanitization

Day 5: API documentation and testing

- Swagger/OpenAPI documentation
- Unit testing
- Integration testing

Phase 3: Web Application (Week 5–6)

Week 5: Frontend Foundation

Day 1: React setup with TypeScript

- Tailwind configuration
- Axios setup
- Folder structure

Day 2: Authentication pages

- Login page
- Google login integration
- Token handling
- Session management

Day 3: Dashboard page

- Overview cards
- Pending evaluations list
- Create Evaluation button

Day 4: Forms Created page

- Table view of created forms
- Edit / Delete / Publish actions
- Search & filter

Day 5: Evaluation answering interface

- Dynamic question rendering
- Submit form
- Confirmation message

Week 6: Complete Web Features

Day 1: My Results page

- Graphs and rating summary
- Feedback comments list
- Export results (optional)

Day 2: My Completed Forms page

- List of submitted evaluations
- Submission timestamps
- View-only mode

Day 3: Settings page

- Update profile
- Change password

- Notification preferences
- Logout

Day 4: Responsive design polish

- Tablet optimization
- Mobile browser support

Day 5: API integration and testing

- End-to-end web testing
- Fix integration issues

Phase 4: Mobile Application (Week 7–8)

Week 7: Android Foundation

Day 1: Android Studio setup and project structure

- Kotlin + Jetpack Compose setup
- Retrofit API configuration
- Room database setup

Day 2: Authentication screens

- Login screen
- Google OAuth integration
- Secure token storage

Day 3: Dashboard screen

- Pending evaluations
- Quick access buttons

Day 4: Evaluation answering screen

- Dynamic Compose UI for questions
- Submit evaluation

Day 5: API service layer

- Retrofit service interfaces
- Repository pattern implementation

Week 8: Complete Mobile App

Day 1: My Results screen

- Rating charts
- Feedback display

Day 2: Forms Created & Completed Forms screens

- Table-style list view
- Edit/View functionality

Day 3: UI polish and animations

- Compose transitions
- Loading indicators

- Error states

Day 4: Testing on emulator/device

- Functional testing
- Performance testing

Day 5: APK generation and documentation

- Signed APK build
- User guide documentation

Phase 5: Integration & Deployment (Week 9–10)

Week 9: Integration Testing

Day 1: End-to-end testing across platforms

- Web + Backend
- Mobile + Backend

Day 2: Bug fixes and optimization

- UI fixes
- Performance improvements

Day 3: Security review

- JWT validation
- OAuth verification

- Access control testing

Day 4: Performance testing

- Load testing
- Database query optimization

Day 5: Documentation updates

- Technical documentation
- User manual

Week 10: Deployment

Day 1: Backend deployment (Railway/Heroku)

- Configure environment variables
- Production database setup

Day 2: Web app deployment (Vercel/Netlify)

- Production build
- Domain configuration

Day 3: Mobile APK distribution

- APK sharing
- Installation guide

Day 4: Final testing

- Smoke testing
- Live bug verification

Day 5: Project submission

- Final report
- Presentation preparation

Milestones

M1 (End Week 2): All design documents complete
M2 (End Week 4): Backend API fully functional
M3 (End Week 6): Web application complete
M4 (End Week 8): Mobile application complete
M5 (End Week 10): Full PeerTayo system deployed and integrated

Critical Path

- Authentication system (Week 3)
- Evaluation Form CRUD (Week 3–4)
- Submission & Results computation (Week 4)
- Web UI integration (Week 5–6)
- Mobile integration (Week 7–8)
- Cross-platform testing (Week 9)

Risk Mitigation

- Start with simplest working evaluation flow
- Test authentication early
- Integrate Web + Backend early (Week 5)
- Keep stable backup branches
- Prioritize core features (Create Form → Submit → View Results) before enhancements