OKR Term-Project Report

Selin Çırak 21011603

Table of Contents

---

1. Overview

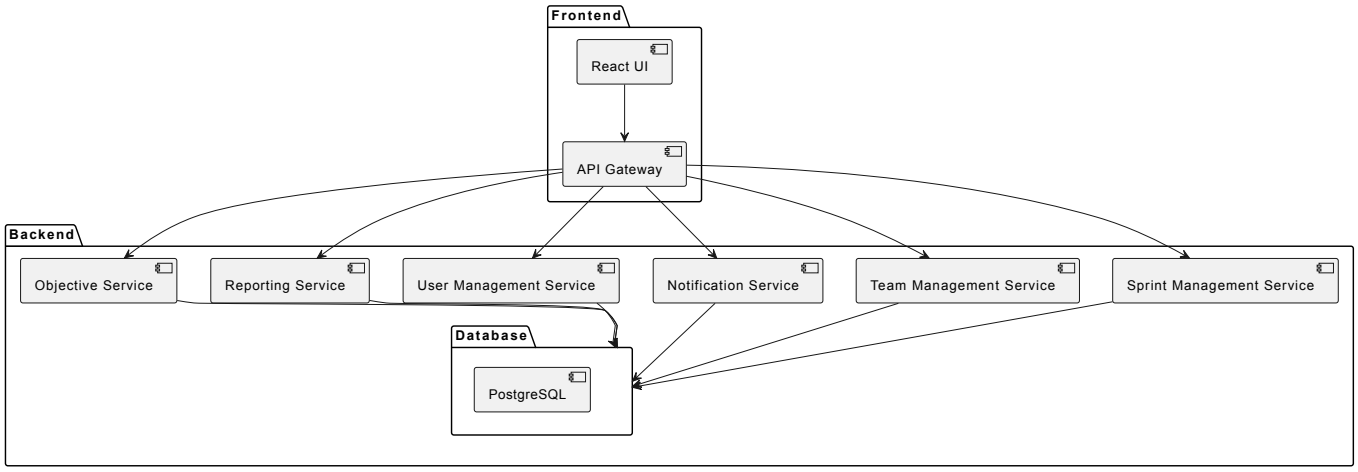
The OKR project is a system designed to help organizations manage their Objectives and Key Results efficiently. The system offers a backend written in Go (using the Fiber framework), a frontend powered by React and Zustand, and Docker for containerization. The backend focuses on handling user management, objectives, key results, notifications, and reporting, while the frontend provides a user-friendly interface for managing and tracking OKRs.

---

2. System Design

2.1 Software Design

The system follows a modular, microservice architecture where each service is responsible for a specific domain, and all services communicate through RESTful APIs. Services are containerized using Docker to allow for easy deployment and scalability.



Backend Modules:

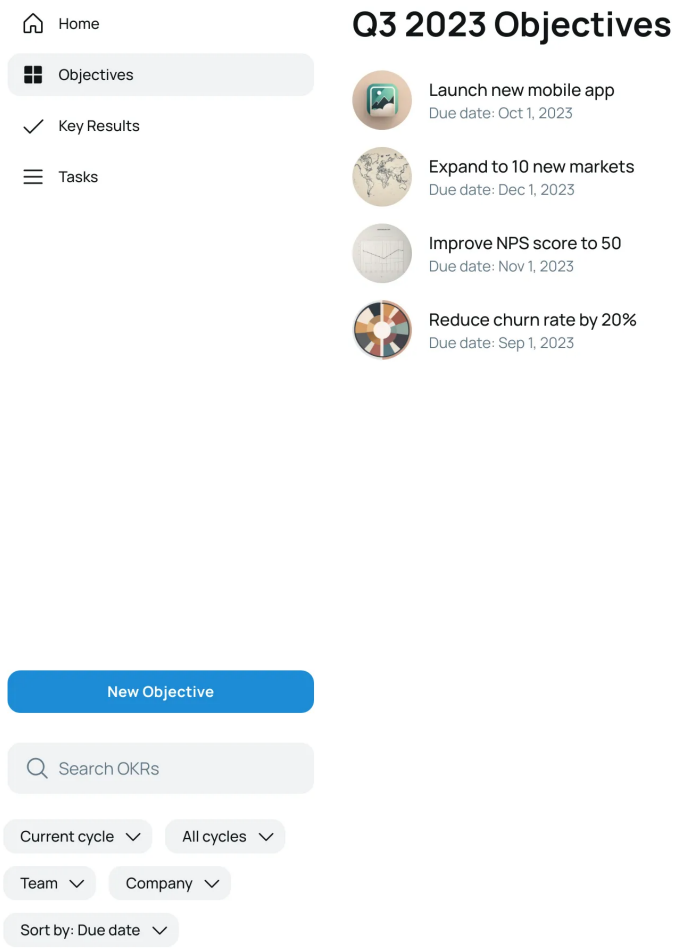
- **User Management Service**: Handles authentication, user roles, and team management.
- **Objective Service**: Manages objectives, key results, and associated comments.
- **Reporting Service**: Generates detailed reports and graphs for insights.
- **Notification Service**: Manages user notifications and system alerts.
- **Team Management Service**: Manages team creation and assignment of team members to objectives.
- **Sprint Management Service**: Handles the management of sprint tasks and assignments.

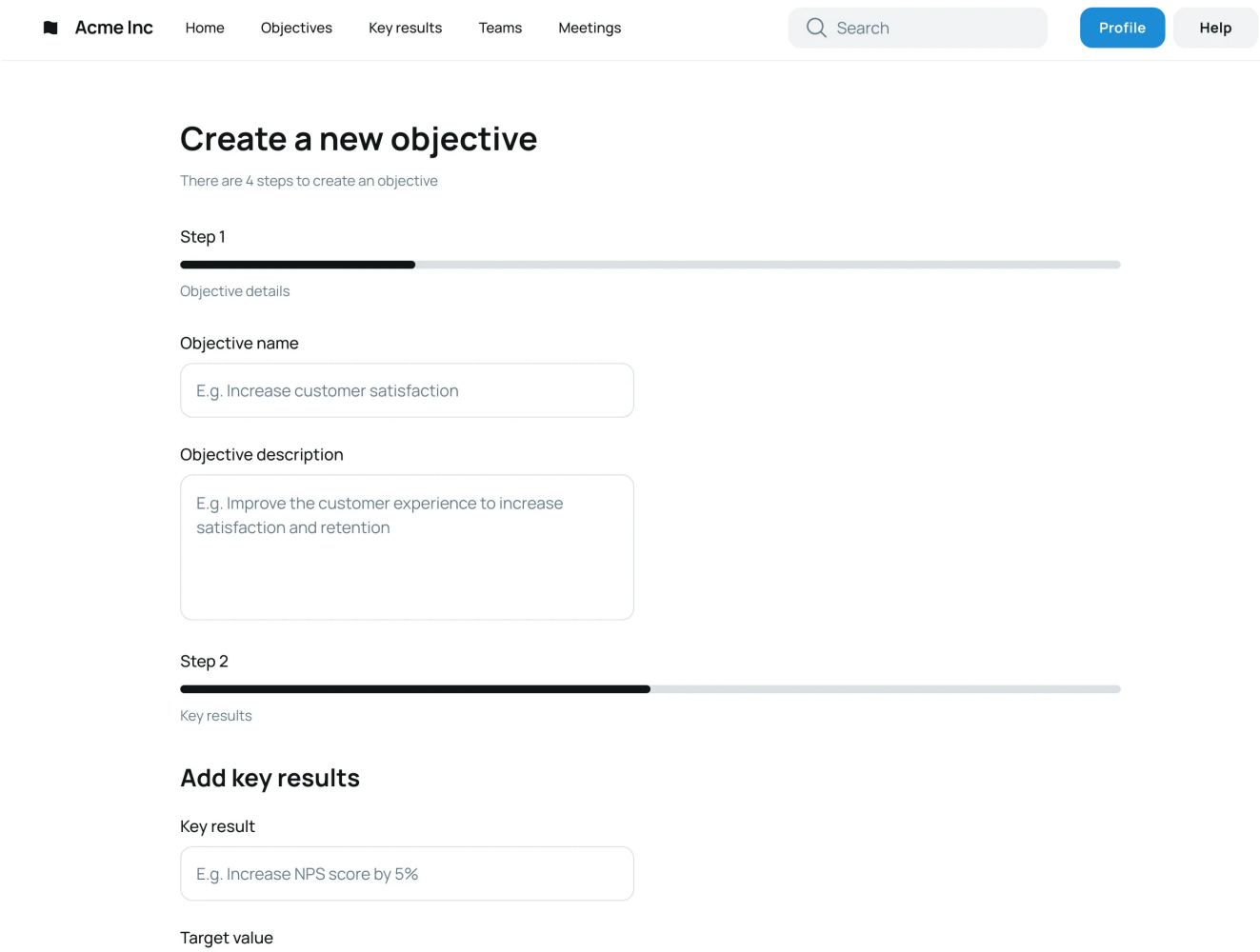
Deployment Workflow:

- Services are deployed in isolated Docker containers.
- The backend services are orchestrated via an **API Gateway** to route requests.
- All data is stored in a PostgreSQL database.

2.2 Interface Design

Screenshots from Figma design sheet:


Home
Objectives
Key Results
Tasks
Q3 2023 Objectives
Launch new mobile app
Due date: Oct 1, 2023
Expand to 10 new markets
Due date: Dec 1, 2023
Improve NPS score to 50
Due date: Nov 1, 2023
Reduce churn rate by 20%
Due date: Sep 1, 2023
New Objective
Search OKRs
Current cycle
All cycles
Team
Company
Sort by: Due date


Acme Inc
Home
Objectives
Key results
Teams
Meetings
Search
Profile
Help
Create a new objective
There are 4 steps to create an objective
Step 1
Objective details
Objective name
E.g. Increase customer satisfaction
Objective description
E.g. Improve the customer experience to increase satisfaction and retention
Step 2
Key results
Add key results
Key result
E.g. Increase NPS score by 5%
Target value

E.g. 5%

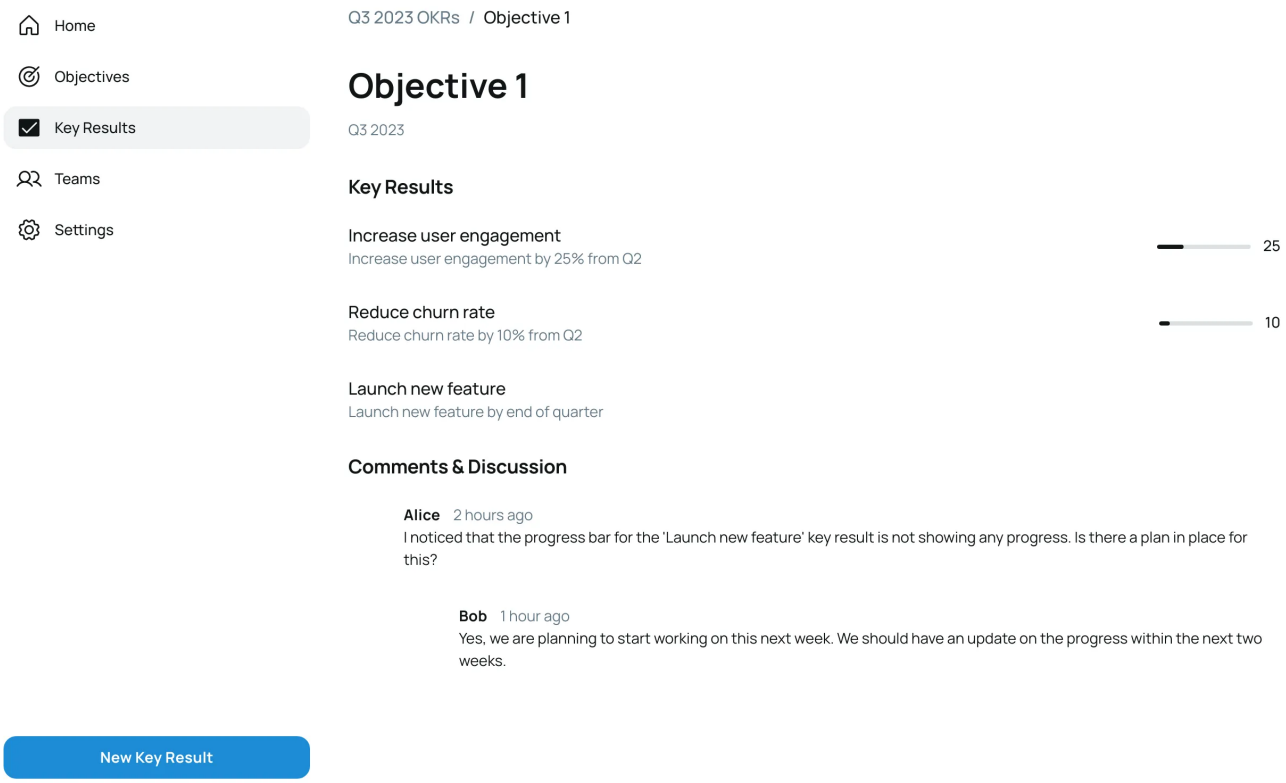
Initial value

E.g. Current NPS score is 40%

- [x] Automatically track this key result
- [ ] Include in parent objective's progress
- [ ] Notify me when this key result is updated

+ Add key result

Next



Q3 OKRs
3/5 Milestones

Overview

Milestones

Labels

Activity

Insights

+ Create Milestone

Q3 OKRs

3/5 Milestones

Q3 Milestones

- ✓ Q3 OKRs
  July 1, 2023
- ✓ Design sprint
  July 5, 2023
- ✓ Cross-functional training
  July 10, 2023
- ✓ Q3 planning
  July 14, 2023
- ✓ Milestone 1
  July 20, 2023

Q3 OKRs

Objective 1: Improve cross-functional collaboration

70%

Objective 2: Speed up design and development processes

90%

Objective 3: Increase customer satisfaction

50%

Q3 Key Results

KR 1: Foster a culture of feedback

80%

KR 2: Reduce time spent on manual tasks

60%

KR 3: Boost our NPS score by 10 points

40%

Growth

- Home
- Objectives
- Key Results
- Teams
- Insights

New OKR

Insights

Company | Engineering | Product

Search for an Objective

Progress

On track	Behind	At risk	Total
220	40	8	268

Progress on OKRs 82%

Heatmap


July 2022
268 OKRs


June 2022
268 OKRs


May 2022
268 OKRs


April 2022
268 OKRs


March 2022
268 OKRs


February 2022
268 OKRs


January 2022
268 OKRs


December 2021
268 OKRs


November 2021
268 OKRs

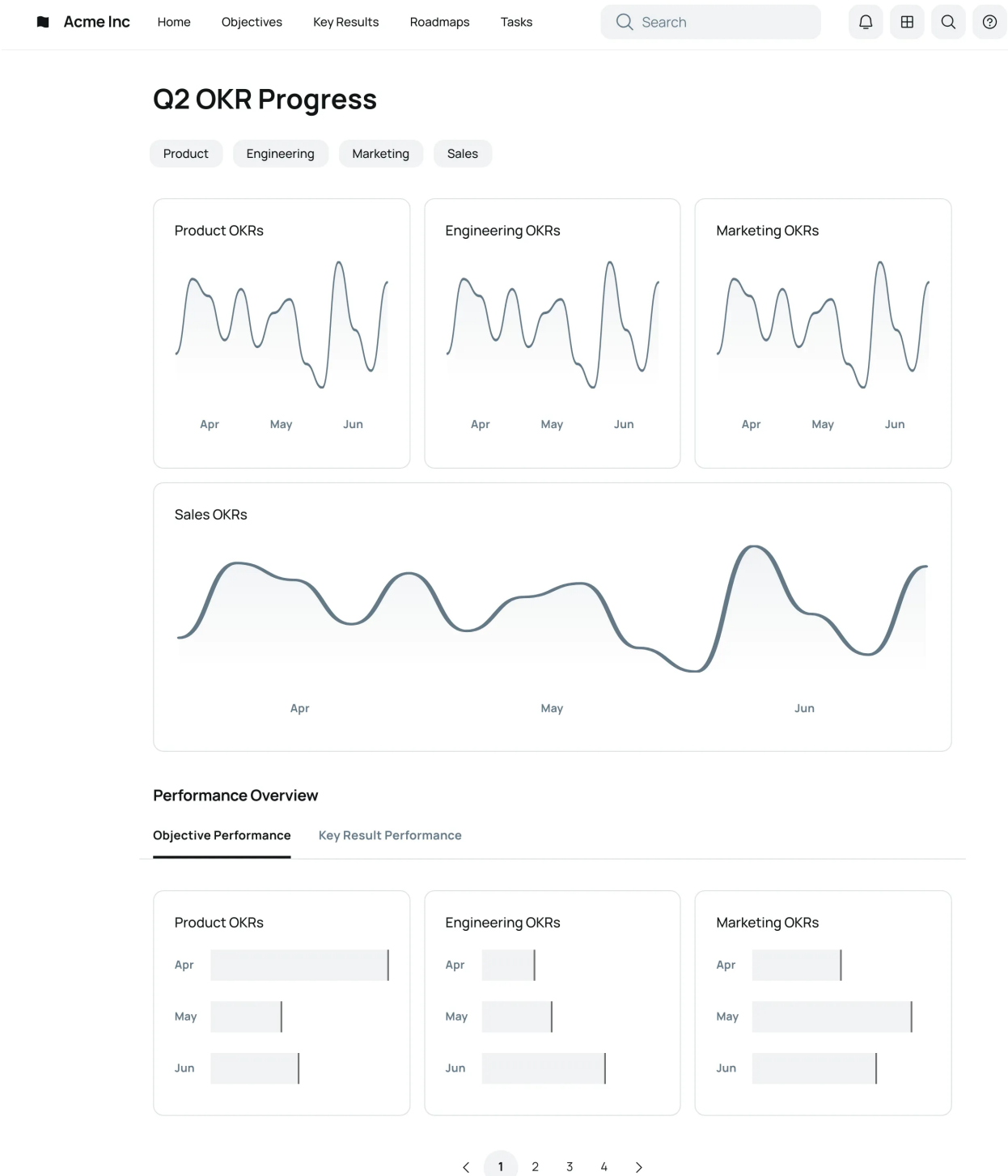

October 2021
268 OKRs


September 2021
268 OKRs


August 2021
268 OKRs

Key Results

Key Result	Objective	Owner	Progress	Target	Start	End
Improve customer satisfaction by 10%	Increase customer satisfaction		70	10%	Jul 1	Sep 30
Launch new feature	Improve product experience		50	100%	Jun 15	Aug 15
Reduce error rate by 20%	Stabilize backend system		90	20%	Jul 5	Oct 1
Increase weekly active users by 15%	Grow user base		60	15%	Jul 10	Sep 30
Improve app rating to 4.5 stars	Enhance user experience		75	4.5	Jul 20	Aug 30
Reduce customer complaints by 25%	Improve support service		80	25%	Jun 25	Sep 15
Achieve 95% code coverage in tests	Enhance software quality		85	95%	Jul 8	Oct 5


Acme Inc
Home
Objectives
Key Results
Roadmaps
Tasks
Search
Q2 OKR Progress
Product
Engineering
Marketing
Sales
Product OKRs
Apr
May
Jun
Engineering OKRs
Apr
May
Jun
Marketing OKRs
Apr
May
Jun
Sales OKRs
Apr
May
Jun
Performance Overview
Objective Performance
Key Result Performance
Product OKRs
Apr
May
Jun
Engineering OKRs
Apr
May
Jun
Marketing OKRs
Apr
May
Jun
1
2
3
4

Acme Co.

- Overview
- Objectives
- Key Results
- Tasks
- Roadmap
- People

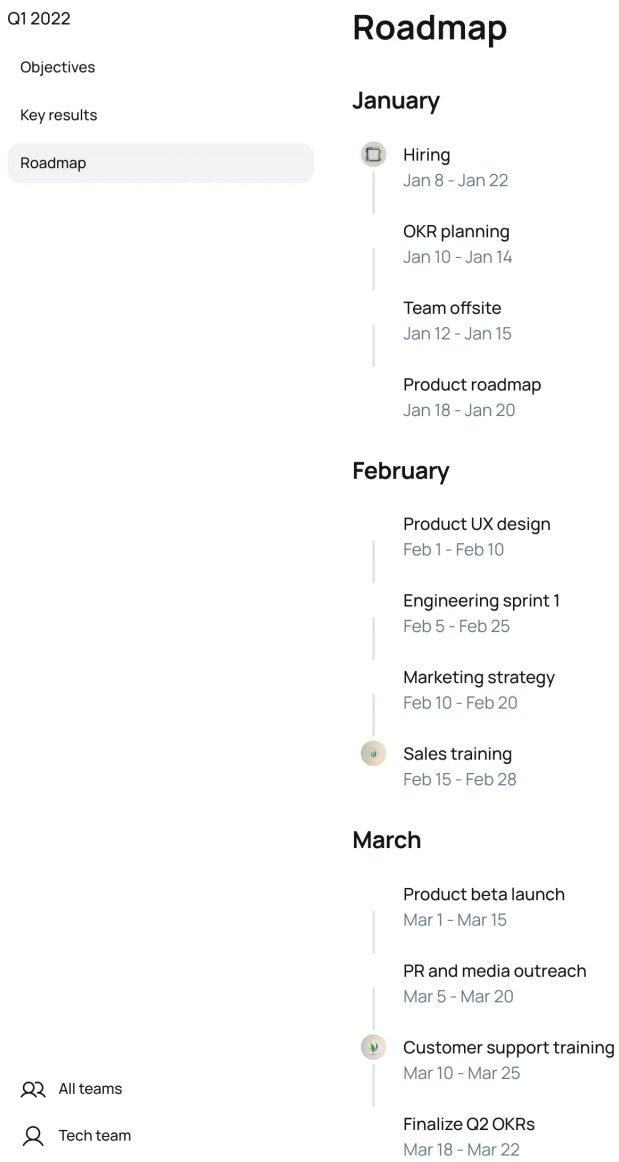
Q3 OKRs

New Objective

All | In Progress | Done

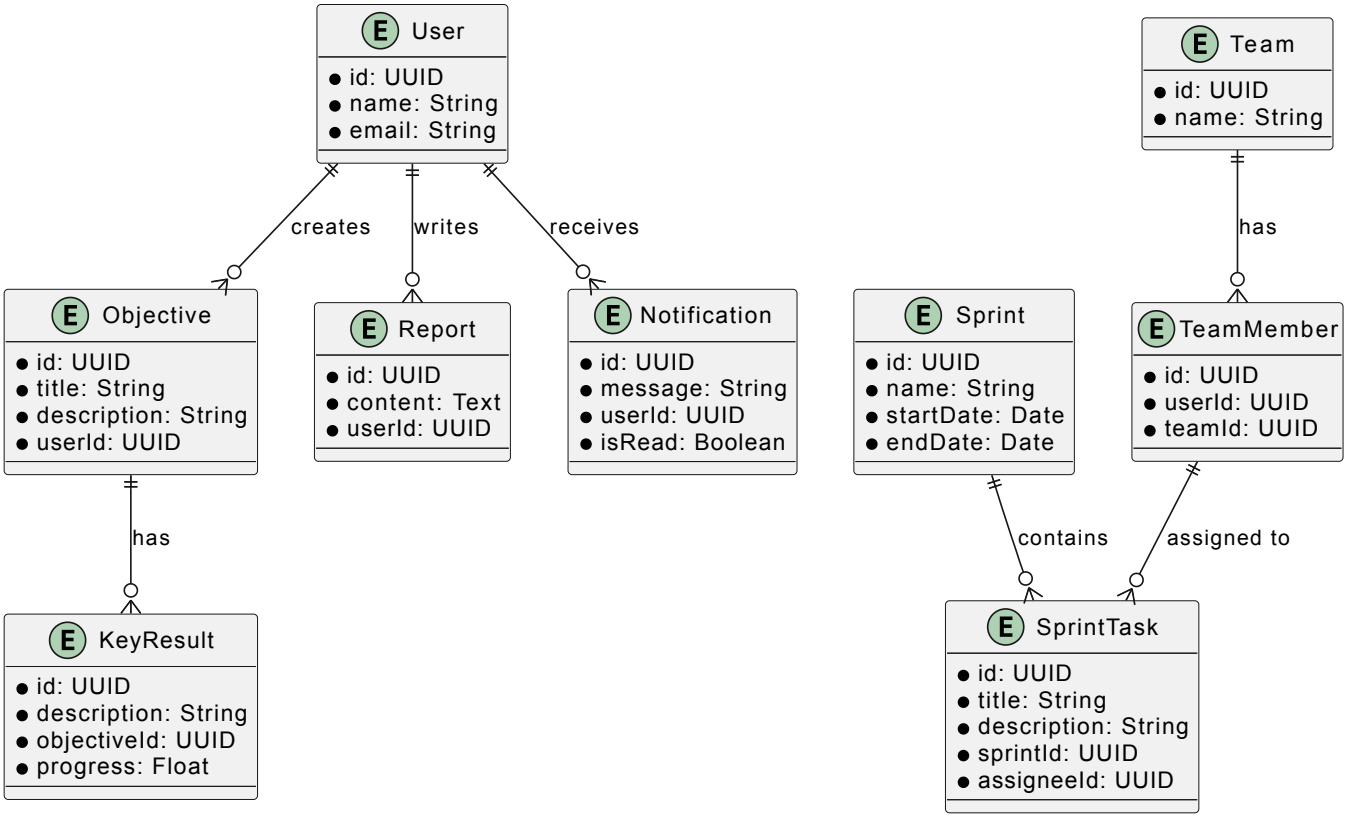
Backlog | Design | Development | Testing

Title	Assignee	Priority	Status	Due Date
Update landing page hero		High	To Do	Aug 15
Implement new checkout flow		High	In Progress	Sep 10
Optimize product images for web		Medium	To Do	Aug 20
Add search feature to knowledge base		Medium	In Progress	Sep 05
Fix responsive layout bug on mobile		Low	To Do	Aug 25
Integrate payment gateway for subscriptions		High	Done	Aug 30
Create user onboarding tutorial video		Low	In Progress	Sep 01
Implement dark mode feature for app UI		Medium	To Do	Aug 22
Improve site performance by optimizing code		High	In Progress	Sep 08
Add multi-language support for product descriptions		Low	To Do	Aug 18



2.3 Database Design

The PostgreSQL database stores all data related to users, objectives, key results, notifications, and more. Below is the updated Entity-Relationship (E-R) diagram:



---

3. Stored Procedures

Updated Stored Procedures

Calculate Objective Progress

```
CREATE OR REPLACE FUNCTION CalculateObjectiveProgress(objective_id UUID)
RETURNS FLOAT AS $$
DECLARE
  total_progress FLOAT;
BEGIN
  SELECT AVG(progress) INTO total_progress
  FROM KeyResult
  WHERE objectiveId = objective_id;

  RETURN total_progress;
END;
$$ LANGUAGE plpgsql;
```

Get Notifications for User

```
CREATE OR REPLACE FUNCTION GetUserNotifications(user_id UUID)
RETURNS TABLE(id UUID, message TEXT, isRead BOOLEAN) AS $$
BEGIN
  RETURN QUERY SELECT id, message, isRead
  FROM Notification
```

```
  WHERE userId = user_id;
END;
$$ LANGUAGE plpgsql;
```

Assign Task to User

```
CREATE OR REPLACE FUNCTION AssignTaskToUser(task_id UUID, user_id UUID)
RETURNS VOID AS $$
BEGIN
  UPDATE SprintTask
  SET assigneeId = user_id
  WHERE id = task_id;
END;
$$ LANGUAGE plpgsql;
```

Create Objective

```
CREATE OR REPLACE FUNCTION CreateObjective(title TEXT, description TEXT,
user_id UUID)
RETURNS UUID AS $$
DECLARE
  objective_id UUID;
BEGIN
  INSERT INTO Objective (title, description, userId)
  VALUES (title, description, user_id)
  RETURNING id INTO objective_id;

  RETURN objective_id;
END;
$$ LANGUAGE plpgsql;
```

Update Objective Progress

```
CREATE OR REPLACE FUNCTION UpdateObjectiveProgress(objective_id UUID)
RETURNS VOID AS $$
DECLARE
  total_progress FLOAT;
BEGIN
  -- Calculate the average progress of all key results related to this
objective
  SELECT AVG(progress) INTO total_progress
  FROM KeyResult
  WHERE objectiveId = objective_id;

  -- Update the objective's progress field
  UPDATE Objective
```

```
  SET progress = total_progress
  WHERE id = objective_id;
END;
$$ LANGUAGE plpgsql;
```

Create Sprint

```
CREATE OR REPLACE FUNCTION CreateSprint(name TEXT, start_date DATE,
end_date DATE)
RETURNS UUID AS $$
DECLARE
  sprint_id UUID;
BEGIN
  INSERT INTO Sprint (name, startDate, endDate)
  VALUES (name, start_date, end_date)
  RETURNING id INTO sprint_id;

  RETURN sprint_id;
END;
$$ LANGUAGE plpgsql;
```

Assign User to Team

```
CREATE OR REPLACE FUNCTION AssignUserToTeam(user_id UUID, team_id UUID)
RETURNS VOID AS $$
BEGIN
  INSERT INTO TeamMember (userId, teamId)
  VALUES (user_id, team_id);
END;
$$ LANGUAGE plpgsql;
```

Update Task Status

```
CREATE OR REPLACE FUNCTION UpdateTaskStatus(task_id UUID, status TEXT)
RETURNS VOID AS $$
BEGIN
  UPDATE SprintTask
  SET status = status
  WHERE id = task_id;
END;
$$ LANGUAGE plpgsql;
```

Mark notification read

```
CREATE OR REPLACE FUNCTION MarkNotificationAsRead(notification_id UUID)
RETURNS VOID AS $$
BEGIN
  UPDATE Notification
  SET isRead = TRUE
  WHERE id = notification_id;
END;
$$ LANGUAGE plpgsql;
```

Delete Objective

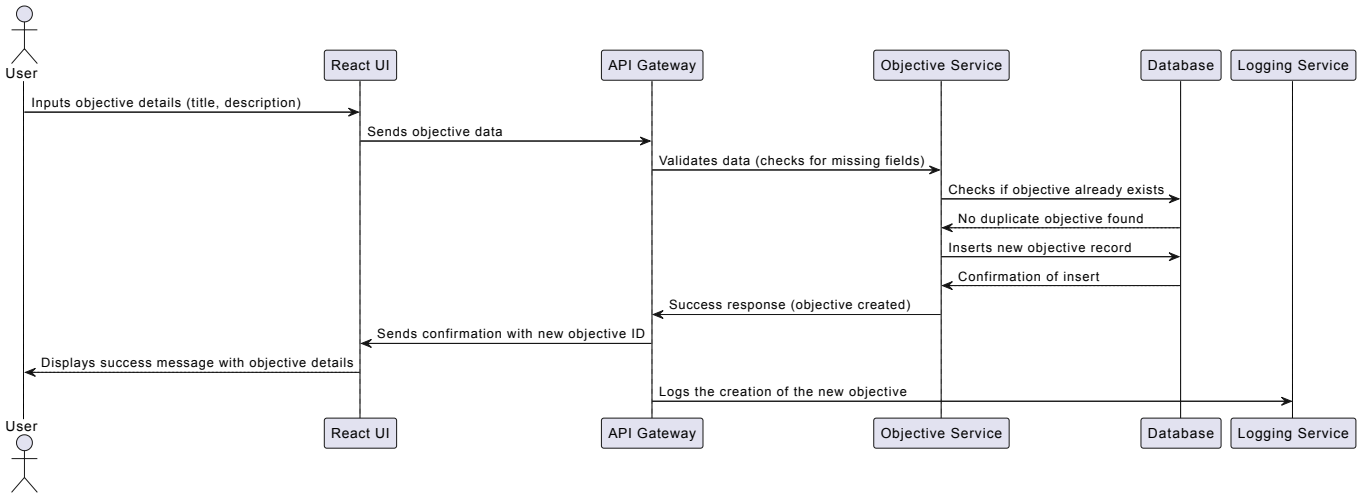
```
CREATE OR REPLACE FUNCTION DeleteObjective(objective_id UUID)
RETURNS VOID AS $$
BEGIN
  -- Delete related key results first
  DELETE FROM KeyResult
  WHERE objectiveId = objective_id;

  -- Then delete the objective itself
  DELETE FROM Objective
  WHERE id = objective_id;
END;
$$ LANGUAGE plpgsql;
```

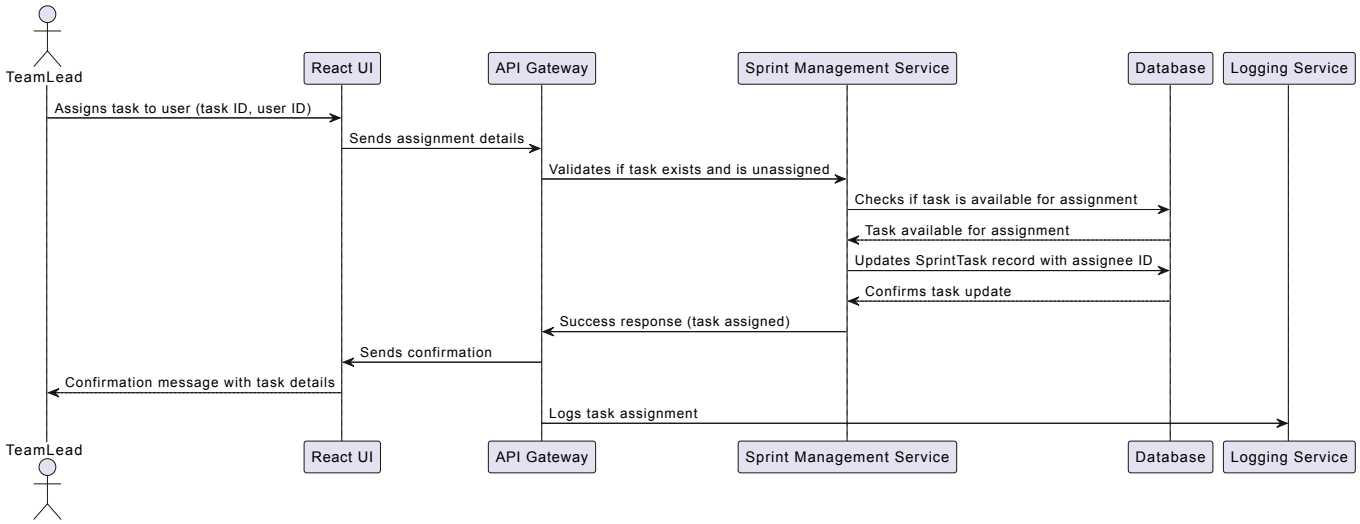
---

4. Use Case Sequence Diagrams

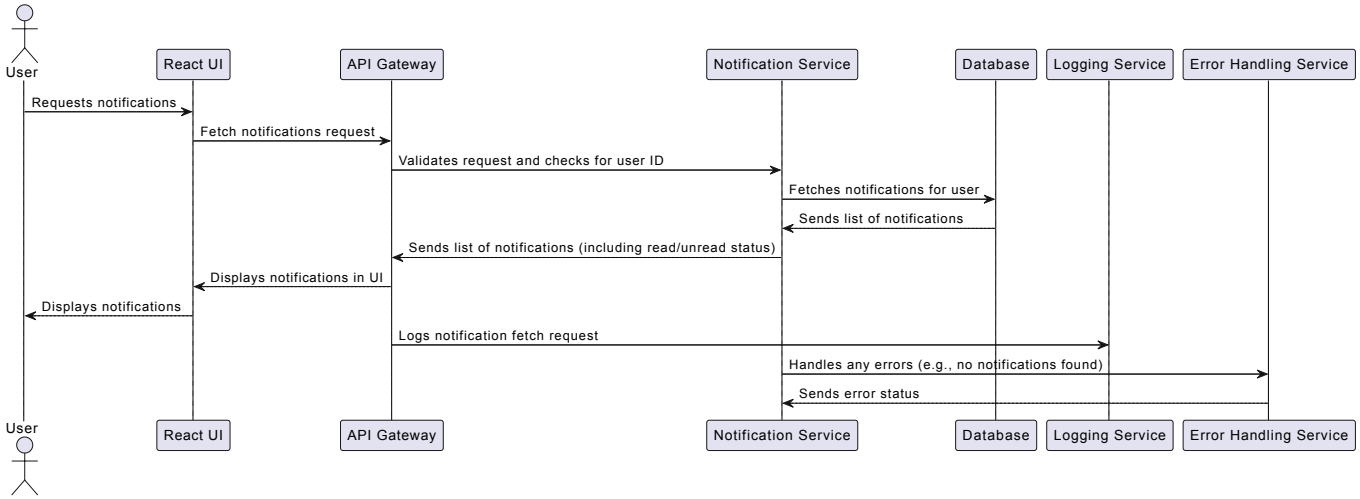
Sequence Diagram: Adding an Objective



Sequence Diagram: Assigning a Task

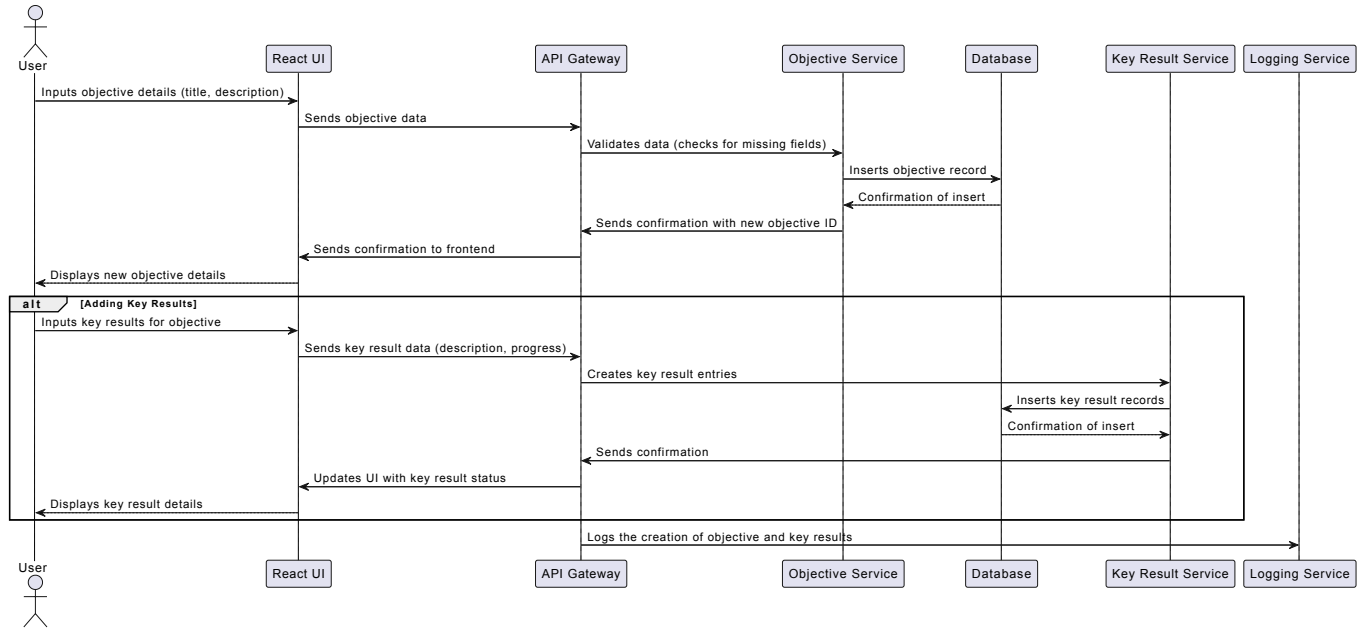


Sequence Diagram: Viewing Notifications



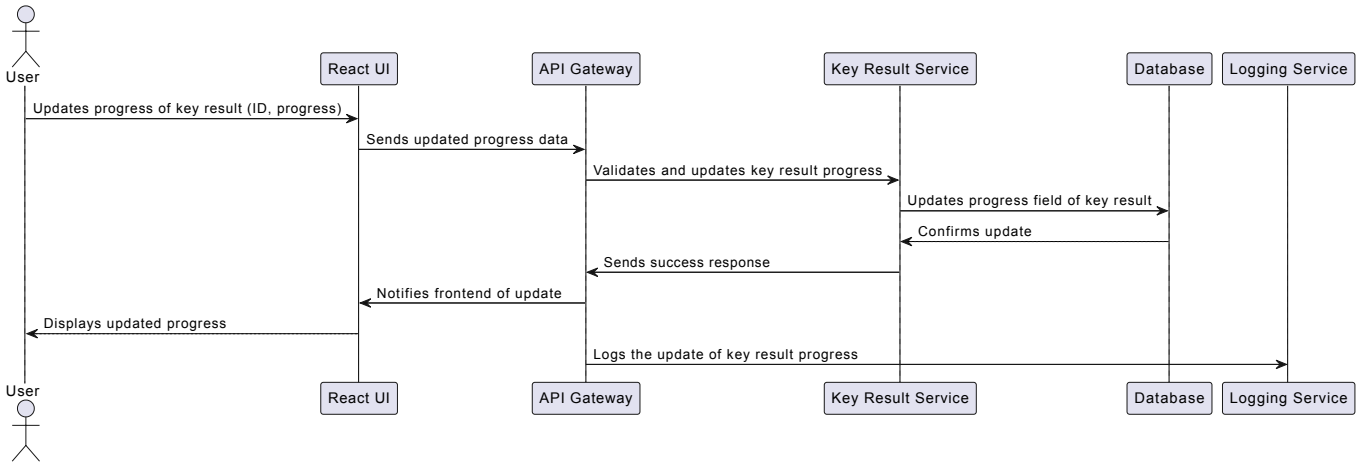
Sequence Diagram 4: Creating an Objective with Key Results

This diagram represents the flow when a user creates an objective and adds key results to it.



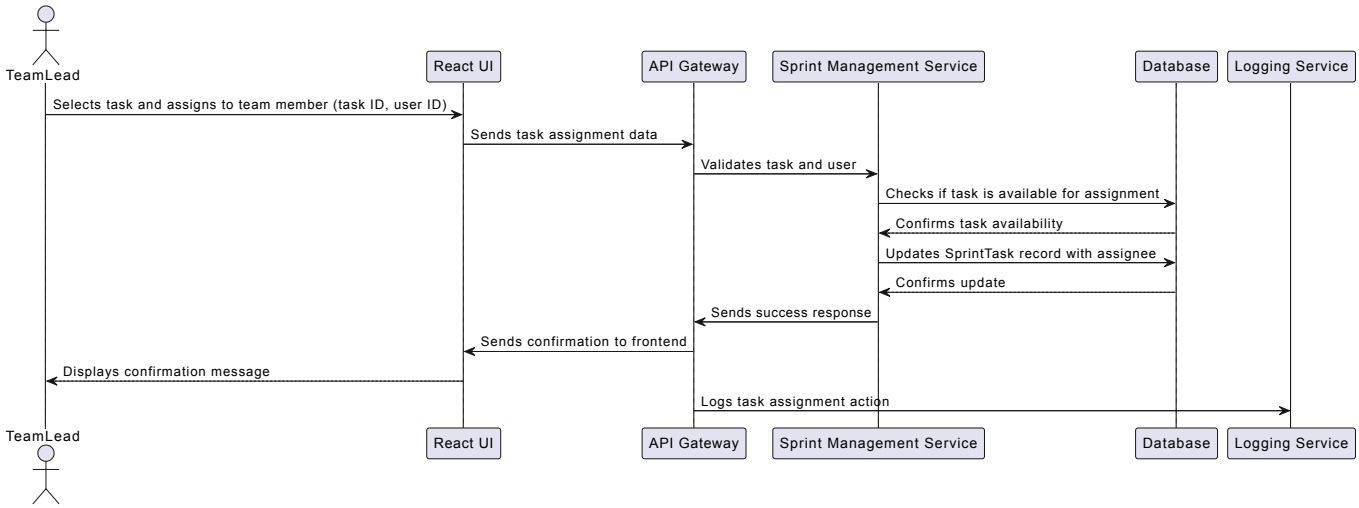
Sequence Diagram 5: Updating the Progress of a Key Result

This sequence demonstrates the flow when a user updates the progress of a key result.



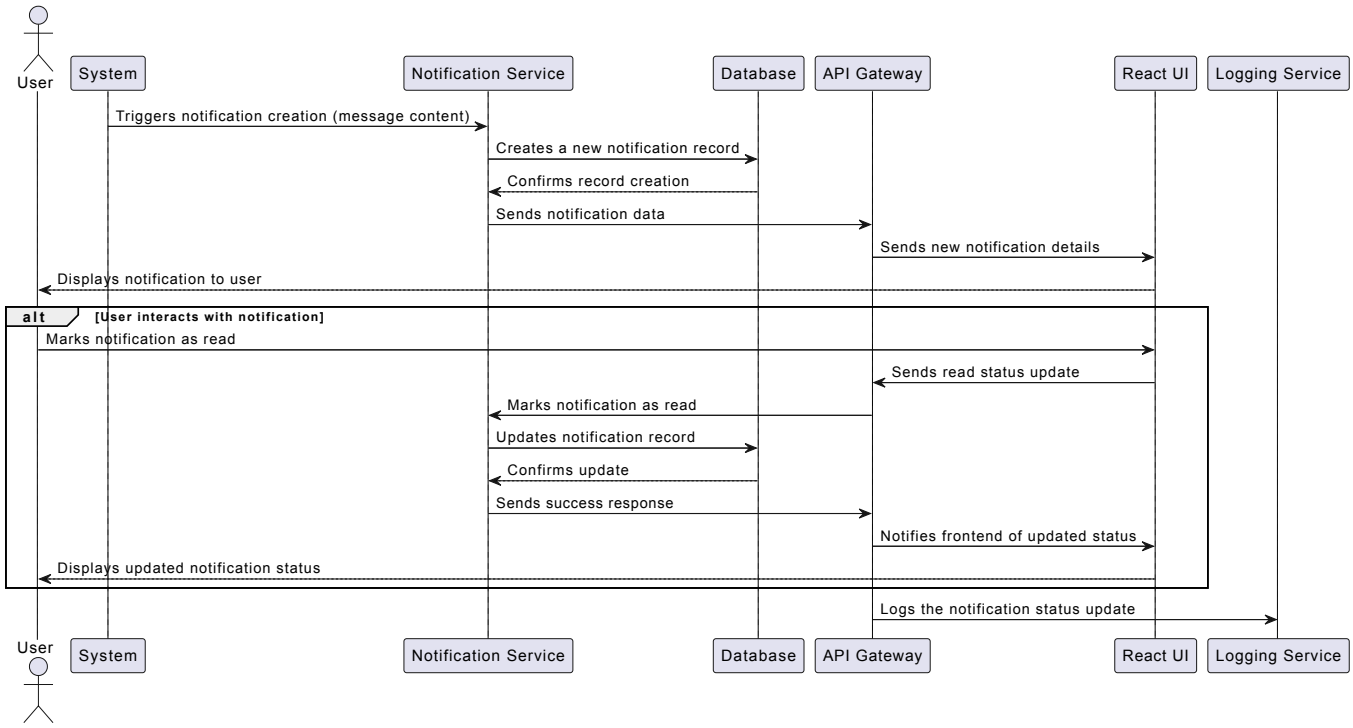
Sequence Diagram 6: Assigning a Sprint Task to a Team Member

This sequence illustrates the process of assigning a sprint task to a team member.



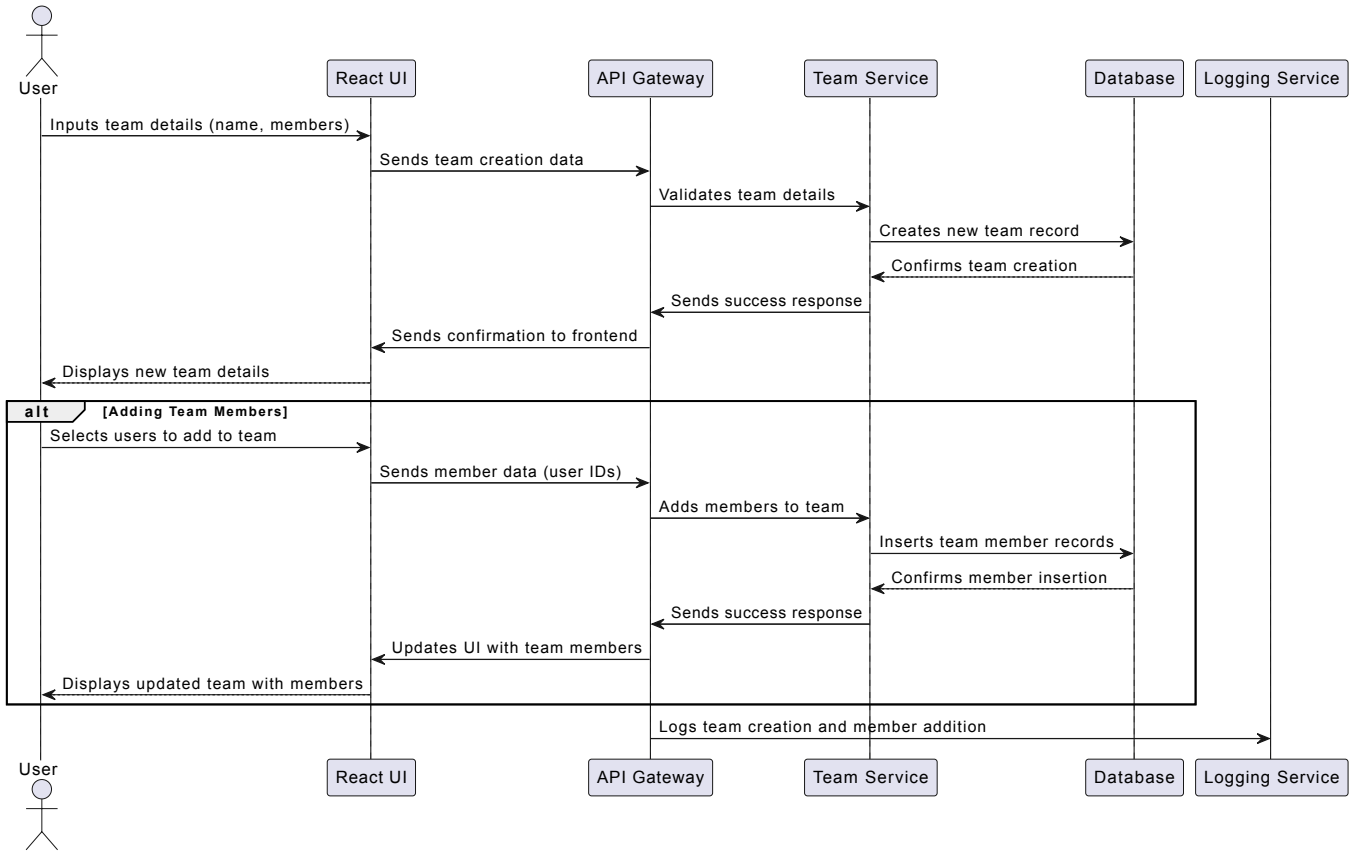
Sequence Diagram 7: Sending Notifications to Users

This sequence diagram shows how the notification system works when a new notification is generated and sent to a user.



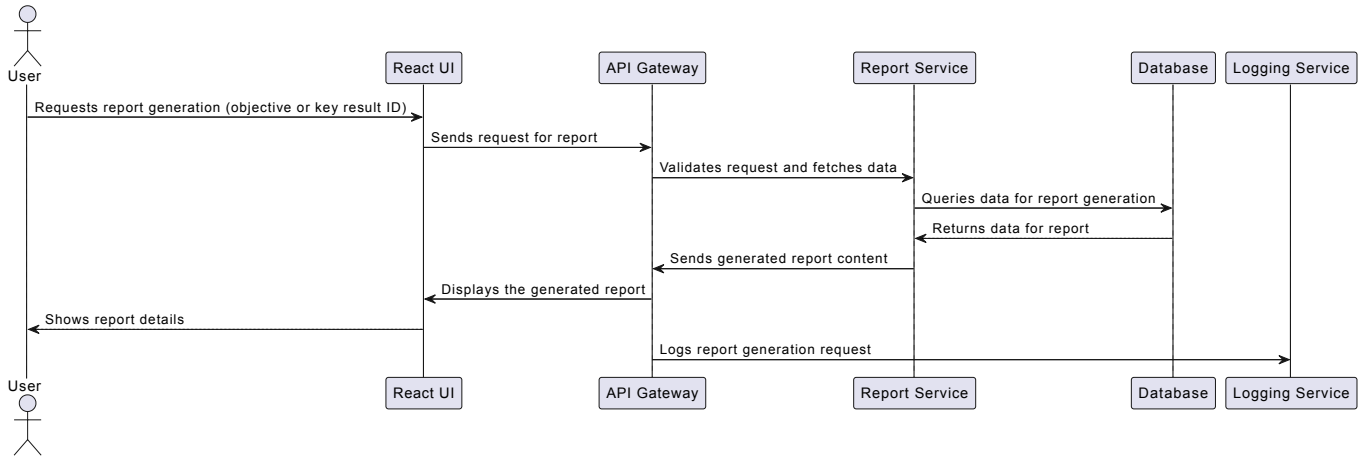
Sequence Diagram 8: Creating a Team and Adding Members

This sequence shows the flow when a team is created and members are assigned to the team.



Sequence Diagram 9: Generating a Report

This diagram illustrates the process for generating a report on the user's objectives or key results.



API Documentation

Below is the comprehensive API documentation for the OKR (Objectives and Key Results) system. This documentation covers various aspects of the system, including authentication, managing objectives, reports, notifications, tasks, and teams.

---

1. Authentication

- **POST /auth/login**

  This endpoint is used for logging in and obtaining an authentication token.

  - **Request Body:**

```
{
    "email": "string",
    "password": "string"
}
```

  - **Response:**

```
{
    "token": "string"
}
```

---

2. Objectives

POST /objectives

This endpoint creates a new objective.

- **Request Body:**

```
{
    "title": "string",
    "description": "string",
    "userId": "UUID"
}
```

- **Response:**

```
{
    "id": "UUID",
    "title": "string"
}
```

GET /objectives

This endpoint retrieves a list of objectives for the authenticated user.

- **Response:**

```
[
    {
        "id": "UUID",
        "title": "string",
        "progress": 0.5
    }
]
```

---

3. Key Results

POST /keyresults

This endpoint allows the creation of a new key result under an objective.

- **Request Body:**

```
{
    "description": "string",
    "objectiveId": "UUID",
    "progress": 0.0
}
```

- **Response:**

```
{
    "id": "UUID",
    "description": "string",
    "progress": 0.0
}
```

GET /keyresults

This endpoint retrieves the list of key results for the objectives.

- **Response:**

```
[
    {
        "id": "UUID",
        "description": "string",
        "progress": 0.0
    }
]
```

PUT /keyresults/{id}/update

This endpoint updates the progress of a key result.

- **Request Body:**

```
{
    "progress": 0.75
}
```

- **Response:**

```
{
    "id": "UUID",
    "description": "string",
    "progress": 0.75
}
```

---

4. Reports

GET /reports

This endpoint retrieves a list of reports written by the user.

- **Response:**

```
[
    {
        "id": "UUID",
        "content": "string"
    }
]
```

---

5. Notifications

GET /notifications

This endpoint retrieves the list of notifications for the authenticated user.

- **Response:**

```
[
    {
        "id": "UUID",
        "message": "string",
        "isRead": false
    }
]
```

POST /notifications/read

This endpoint marks a notification as read.

- **Request Body:**

```
{
    "id": "UUID"
}
```

- **Response:**

```
{
    "status": "success"
}
```

---

6. Tasks and Sprints

POST /tasks/assign

This endpoint assigns a sprint task to a user (team member).

- **Request Body:**

```
{
    "taskId": "UUID",
    "userId": "UUID"
}
```

- **Response:**

```
{
    "status": "success"
}
```

---

7. Teams

GET /teams

This endpoint retrieves the list of teams in the system.

- **Response:**

```
[
    {
        "id": "UUID",
        "name": "Team Name"
    }
]
```

POST /teams

This endpoint creates a new team.

- **Request Body:**

```
{
    "name": "string"
}
```

- **Response:**

```
{
    "id": "UUID",
    "name": "string"
}
```

POST /teams/{teamId}/members

This endpoint adds a member to an existing team.

- **Request Body:**

```
{
    "userId": "UUID"
}
```

- **Response:**

```
{
    "status": "success"
}
```

---

8. Sprints

GET /sprints

This endpoint retrieves all the sprints.

- **Response:**

```
[
    {
        "id": "UUID",
        "name": "Sprint 1",
        "startDate": "2024-01-01",
        "endDate": "2024-01-15"
    }
]
```

POST /sprints

This endpoint creates a new sprint.

- **Request Body:**

```
{
    "name": "Sprint 2",
    "startDate": "2024-02-01",
    "endDate": "2024-02-15"
}
```

- **Response:**

```
{
    "id": "UUID",
    "name": "Sprint 2",
    "startDate": "2024-02-01",
    "endDate": "2024-02-15"
}
```

Explanation of API Endpoints:

- **Authentication Endpoints:** Used for logging in and obtaining a token to interact with other secured endpoints.
- **Objective and Key Result Endpoints:** Allow users to create and manage their objectives and associated key results.
- **Report Endpoints:** Enable users to view and generate reports on their OKRs.
- **Notification Endpoints:** Used to get and mark notifications as read, keeping the user informed of important updates.
- **Task Assignment and Sprint Endpoints:** Facilitate managing tasks within sprints, including assigning them to team members.
- **Team Endpoints:** Enable team management, allowing users to create teams and add members.
- **Sprint Endpoints:** Allow users to create and manage sprints that organize and track the progress of work.
