Title of the Project: MiniTrails

I. Introduction:

We chose this project because the ability to manage thousands of trails around the world is important to fostering a community around hiking. This application makes it easier for experienced hikers to learn more about their next hike, and see if the hike fits with their own ability. We chose this domain because of the wealth of community sourced data. It is important for users to be able to access the information in a quick and user-friendly way, and be able to contribute to the community of trail hiking data.

This application allows users to view user submitted data on hikes. This includes the location, Length of the hike, and the elevation change required to complete the hike. Various statistics on the trails such as ratings and difficulty allow the application to surface recommendations to hikers to help them find their next hike based on what is best suited to them. This app also allows users to create their own profiles and record their own hikes, both as a history of their self-improvement, and to contribute to the community for the next user to find the best next hike. Users can also join clubs to foster a social community around the sport of hiking.

The report is organized into the steps required to carry out the development of a new database focused application, from design of the database, to implementation and testing. The general organization of roles and responsibilities were Kylie (Frontend developer) Josiah (Backend developer) and Alex (Backend developer). Kylie created the frontend display to enable users to query the backend database, Alex organized and cleaned the community sourced data into our app database schema, and Josiah generated data to supplement the community sourced data for ease of use.

II. Our Implementation

- Description of the system architecture

  Our database uses Microsoft SQL and our website is written in Python with user interface components built with React. The database consists of seven tables, five primary tables storing data and two to facilitate many to many joins between the tables. The application has SQL queries and stored procedures that are tied to interactions the user can take on the website to trigger data pulls from the database and data loads into it.

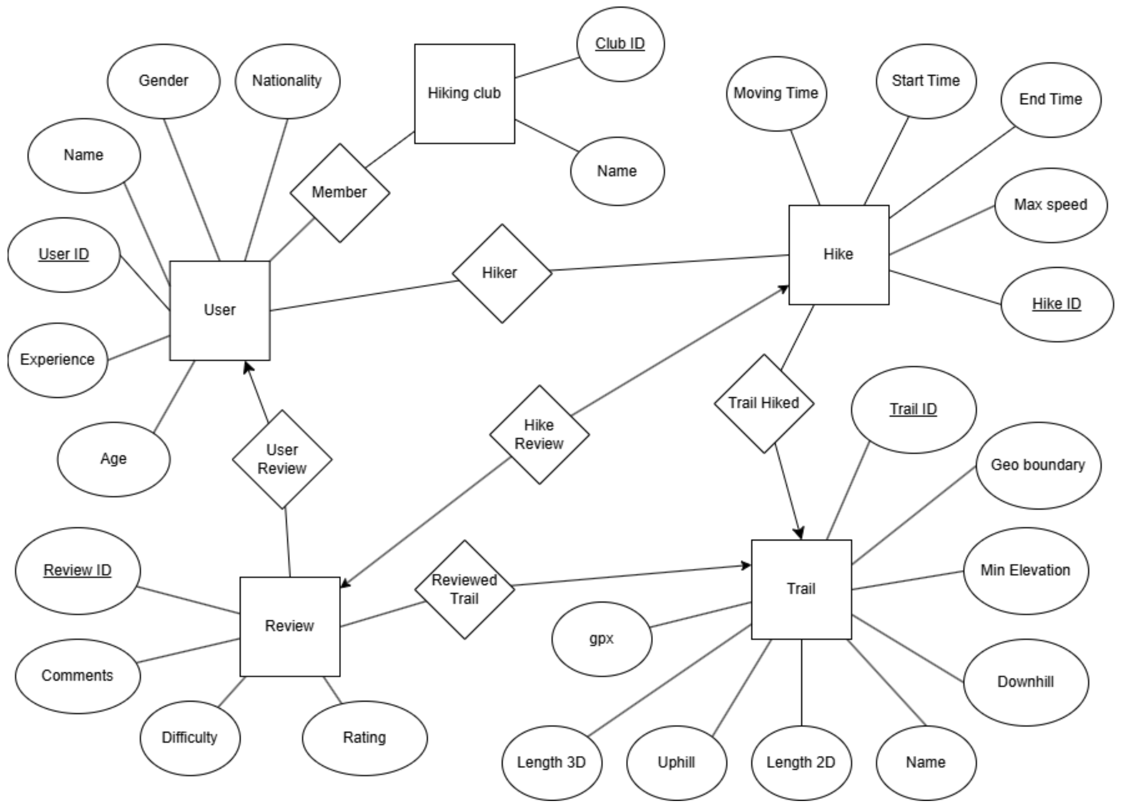
- Description of the dataset

Our data set came from https://www.kaggle.com/datasets/roccoli/gpx-hike-tracks/data and is about hiking, contained 17 columns:

- Id - the dataset assigned identifier
- length_3d - the numeric distance in meters of the trail, including both vertical and horizontal distance
- user - the user name of the user associated with the data row
- start_time - data and time that they began the hike
- max_elevation - max elevation the trail reaches, in meters
- bounds - coordinates of the edges of the trail
- uphill - the vertical distance travel upwards during the hike, in meters
- moving_time - number of seconds the hike took, calculated by finding the difference between their start and end time
- end_time - data and time that they ended the hike
- max_speed - the maximum speed hit while hiking the trail, in kilometers per hour
- gpx - gpx encoded GPS data of the trail's location
- difficulty - rated difficulty of the trail
- min_elevation - min elevation the trail reaches, in meters
- url - if this row's data is scrapped from the internet, the url it came from
- downhill - the vertical distance travel downwards during the hike, in meters
- name - name of the hike
- length_2d - the numeric distance in meters of the trail, including only horizontal distance

We wanted our app to be more than just a repository of hiking data so we generated some additional data of hiking reviews and user demographics so we could simulate the interactions that users would have with our site.

- ER diagram



- Relational model:
  - User(UserID,Name,Gender,Nationality,Age,Experience)
    Keys: UserID is the primary key
  - Review(ReviewID,Comments,Difficulty,Rating,UserID,TrailID)
    Keys: ReviewID [primary key]
  - HikingClub(ClubID,Name)
    Keys: ClubID [primary key]
  - Hike(HikeID,MaxSpeed,EndTime,StartTime,MovingTime,TrailID)
    Keys: HikeID [primary key]
  - Trail(TrailID,GeoBoundary,MinElevation,Downhill,Name,Length2D,Uphill,Length3D,Gpx)
    Keys: TrailID [primary key]
  - Member(UserID,ClubID)
    Keys: (UserID,ClubID) [primary key]
  - Hiker(UserID,HikeID)
    Keys: (UserID,HikeID) [primary key]

- Implementation:
  We built a website that a user would interact with that contains many graphs and tables that query our database and has various data entry fields where they can enter data that gets stored into the database.

We built 7 tables in the database to hold the data, below is the create table statement for each their schemas:

-
  ```
  CREATE TABLE Users (
              UserID BIGINT IDENTITY(1, 1) PRIMARY KEY
              ,Name VARCHAR(255)
              ,Gender VARCHAR(255)
              ,Nationality VARCHAR(255)
              ,Age INT
              ,Experience VARCHAR(255));
  ```

-
  ```
  CREATE TABLE Trail (
              TrailID BIGINT IDENTITY(1, 1) PRIMARY KEY
              ,GeoBoundary VARCHAR(4000)
              ,MinElevation FLOAT
              ,Downhill FLOAT
              ,Name VARCHAR(255)
              ,Length2D FLOAT
              ,Uphill FLOAT
              ,Length3D FLOAT
              ,gpx NVARCHAR(MAX));
  ```

-
  ```
  CREATE TABLE Review (
              ReviewID BIGINT IDENTITY(1, 1) PRIMARY KEY
              ,Comments VARCHAR(4000)
              ,Difficulty VARCHAR(255)
              ,Rating INT
              ,UserID BIGINT
              ,FOREIGN KEY (UserID) REFERENCES Users(UserID)
              ,TrailID BIGINT
              ,FOREIGN KEY (TrailID) REFERENCES Trail(TrailID));
  ```

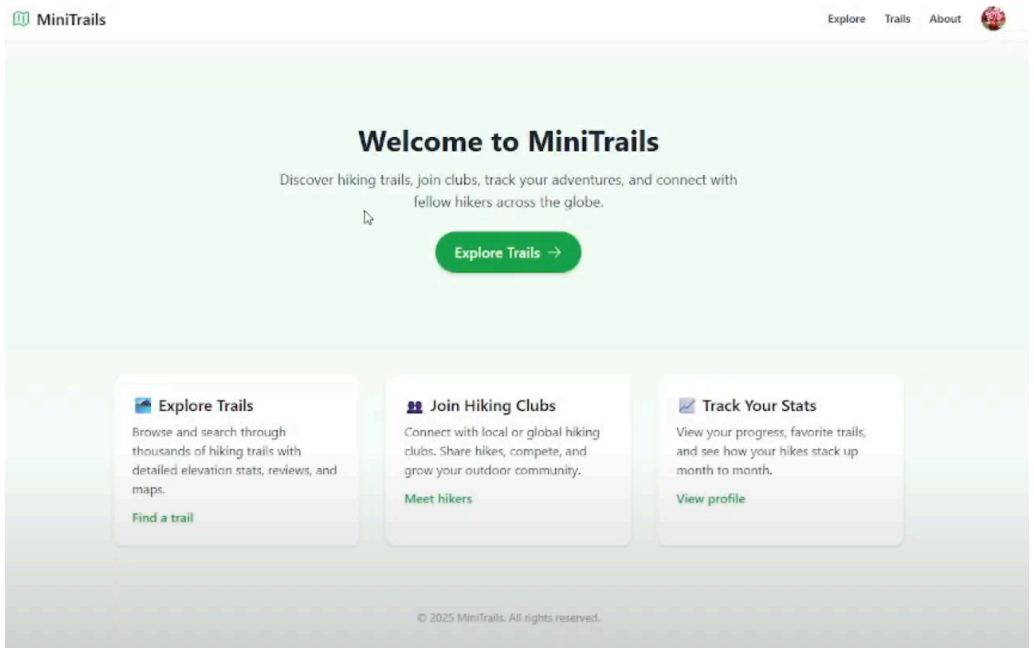
-
  ```
  CREATE TABLE Hike (
              HikeID BIGINT IDENTITY(1, 1) PRIMARY KEY
              ,MaxSpeed FLOAT
              ,EndTime DATETIME2
              ,StartTime DATETIME2
              ,MovingTime FLOAT
              ,TrailID BIGINT
              ,FOREIGN KEY (TrailID) REFERENCES Trail(TrailID));
  ```

-
  ```
  CREATE TABLE HikingClub (
              ClubID BIGINT IDENTITY(1, 1) PRIMARY KEY
              ,Name VARCHAR(255));
  ```

- CREATE TABLE Member (
  UserID BIGINT
  ,ClubID BIGINT
  ,PRIMARY KEY (USERID, ClubID)
  ,FOREIGN KEY (UserID) REFERENCES Users(UserID)
  ,FOREIGN KEY (ClubID) REFERENCES HikingClub(ClubID));

- CREATE TABLE Hiker (
  UserID BIGINT
  ,HikeID BIGINT
  ,PRIMARY KEY (UserID, HikeID)
  ,FOREIGN KEY (UserID) REFERENCES Users(UserID)
  ,FOREIGN KEY (HikeID) REFERENCES Hike(HikeID));

When you first open the website, you see our main page which prompts the user to explore trails, meet other hikers, or view their profile to make changes to their account.



Welcome to MiniTrails

Discover hiking trails, join clubs, track your adventures, and connect with fellow hikers across the globe.

Explore Trails

Browse and search through thousands of hiking trails with detailed elevation stats, reviews, and maps.

Join Hiking Clubs

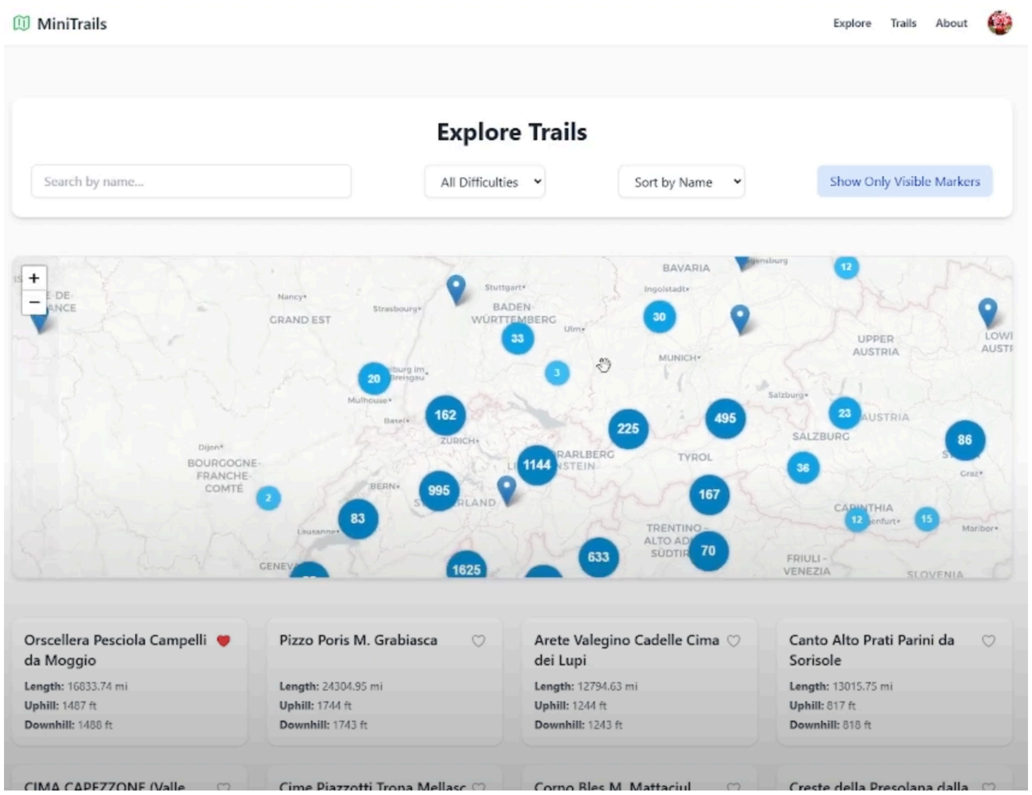
Connect with local or global hiking clubs. Share hikes, compete, and grow your outdoor community.

Track Your Stats

View your progress, favorite trails, and see how your hikes stack up month to month.

© 2025 MiniTrails. All rights reserved.

The first thing on the trails page there's a world map which has pins for the location of each hike in our database. Below that graph it lists all trails shown on the map. You can filter what's shown using the difficulty and name search filters. You can also choose the listing order of trails to be by name, length, or difficulty.


MiniTrails

Explore Trails About

Explore Trails

Search by name... | All Difficulties | Sort by Name | Show Only Visible Markers

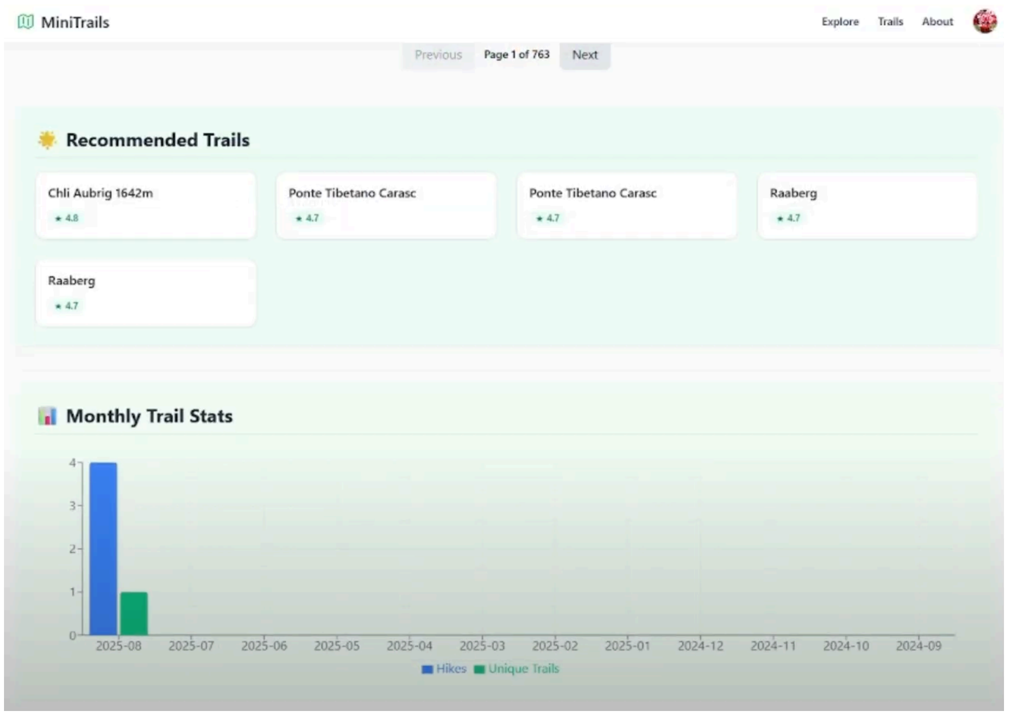
Orscellera Pesciola Campelli da Moggio
Length: 16833.74 mi
Uphill: 1487 ft
Downhill: 1488 ft

Pizzo Poris M. Grabiasca
Length: 24304.95 mi
Uphill: 1744 ft
Downhill: 1743 ft

Arete Valegino Cadelle Cima dei Lupi
Length: 12794.63 mi
Uphill: 1244 ft
Downhill: 1243 ft

Canto Alto Prati Parini da Sorisole
Length: 13015.75 mi
Uphill: 817 ft
Downhill: 818 ft

Below that on the same page are a bunch of graphs and tables. The first is the top 5 recommended hikes for you which is determined by finding the highest rated hikes that the current user has not already completed. Next is a graph of completed hikes per month for the whole database.


MiniTrails

Explore Trails About

Previous Page 1 of 763 Next

Recommended Trails

Chli Aubrig 1642m ★ 4.8

Ponte Tibetano Carasc ★ 4.7

Ponte Tibetano Carasc ★ 4.7

Raaberg ★ 4.7

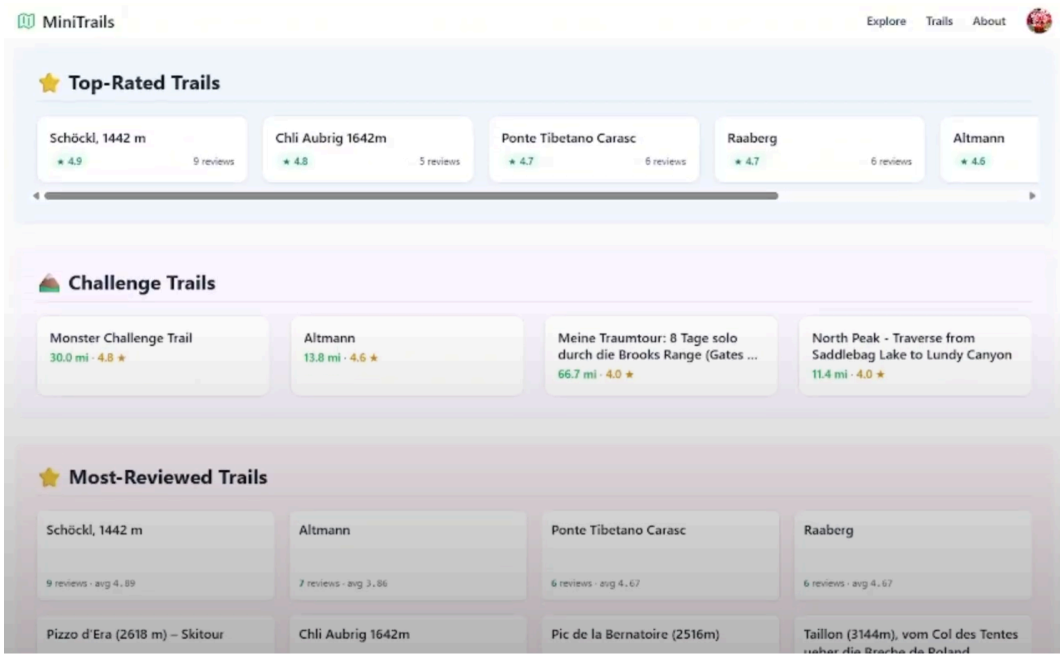
Raaberg ★ 4.7

Monthly Trail Stats

```sql
CREATE   PROC dbo.usp_GetRecommendations
  @UserID      INT,
  @Top         INT = 5,
  @MinReviews  INT = 3
AS
BEGIN
  SET NOCOUNT ON;

  WITH RatingAgg AS (
    SELECT
      R.TrailID,
      AVG(CAST(R.Rating AS FLOAT)) AS avgRating,
      COUNT(*)                     AS numReviews
    FROM dbo.Review R
    GROUP BY R.TrailID
  ),
  Qualified AS (
    SELECT
      T.TrailID,
      T.Name,
      RA.avgRating,
      RA.numReviews
    FROM dbo.Trail T
    JOIN RatingAgg RA ON RA.TrailID = T.TrailID
    WHERE RA.numReviews >= @MinReviews
  ),
  Dedup AS (
    SELECT *,
           ROW_NUMBER() OVER (
             PARTITION BY LTRIM(RTRIM(LOWER(Name)))
             ORDER BY avgRating DESC, numReviews DESC, TrailID
           ) AS rn
    FROM Qualified
  )
  SELECT TOP (@Top)
         TrailID   AS trailId,
         Name      AS name,
         CAST(ROUND(avgRating,2) AS DECIMAL(4,2)) AS avgRating
  FROM Dedup
  WHERE rn = 1
  ORDER BY avgRating DESC, name;
END
```

```
/* Monthly hike counts in the past 12 months (L2) */
SELECT  FORMAT(HI.StartTime, 'yyyy-MM') AS YearMonth,
        COUNT(*)                AS NumHikes
FROM    dbo.Hike HI
WHERE   HI.StartTime >= DATEADD(MONTH, -12, SYSUTCDATETIME())
GROUP BY FORMAT(HI.StartTime, 'yyyy-MM')
ORDER BY YearMonth;
GO
```

After that are three components for the top rated trails, most challenging, and most reviewed trails.



Top-Rated Trails

Schöckl, 1442 m ★ 4.9 9 reviews
Chli Aubrig 1642m ★ 4.8 5 reviews
Ponte Tibetano Carasc ★ 4.7 6 reviews
Raaberg ★ 4.7 6 reviews
Altmann ★ 4.6

Challenge Trails

Monster Challenge Trail 30.0 mi · 4.8 ★
Altmann 13.8 mi · 4.6 ★
Meine Traumtour: 8 Tage solo durch die Brooks Range (Gates ... 66.7 mi · 4.0 ★
North Peak - Traverse from Saddlebag Lake to Lundy Canyon 11.4 mi · 4.0 ★

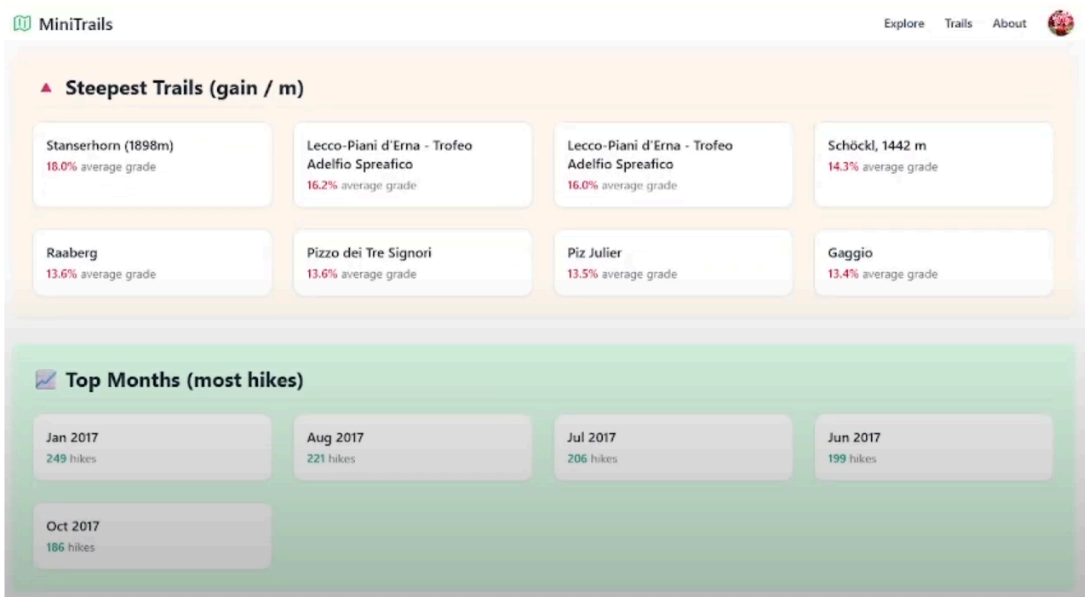
Most-Reviewed Trails

Schöckl, 1442 m 9 reviews · avg 4.89
Altmann 7 reviews · avg 3.86
Ponte Tibetano Carasc 6 reviews · avg 4.67
Raaberg 6 reviews · avg 4.67
Pizzo d'Era (2618 m) – Skitour
Chli Aubrig 1642m
Pic de la Bernatoire (2516m)
Taillon (3144m), vom Col des Tentes ueber die Breche de Roland

```
/* Top Rated and Most-Reviewed Trails */
SELECT  T.TrailID,
        T.Name,
        AVG(R.Rating)     AS AvgRating,
        COUNT(*)          AS NumReviews
FROM    dbo.Trail  AS T
JOIN    dbo.Review AS R ON R.TrailID = T.TrailID
GROUP BY T.TrailID, T.Name
HAVING   COUNT(*) >= 5
ORDER BY AvgRating DESC;
GO
```

```
/* Challenge Trails - Long (>10 mi) & highly-rated (>4) trails*/
SELECT T.TrailID,
      T.Name,
      T.Length3D,
      AR.AvgRating
FROM  ( SELECT TrailID, AVG(Rating) AS AvgRating
       FROM dbo.Review
       GROUP BY TrailID ) AS AR
JOIN  dbo.Trail AS T ON T.TrailID = AR.TrailID
WHERE T.Length3D > 16093.44   -- 10 miles in metres
  AND AR.AvgRating > 4.0;
GO
```

Then is a component for the steepest trails which includes the trail name and its average grade. Finally, there is a component that lists out the Top months of hikes with the number of hikes that occurred during that month.



Steepest Trails (gain / m)

Stanserhorn (1898m)
18.0% average grade

Lecco-Piani d'Erna - Trofeo Adelfio Spreafico
16.2% average grade

Lecco-Piani d'Erna - Trofeo Adelfio Spreafico
16.0% average grade

Schöckl, 1442 m
14.3% average grade

Raaberg
13.6% average grade

Pizzo dei Tre Signori
13.6% average grade

Piz Julier
13.5% average grade

Gaggio
13.4% average grade

Top Months (most hikes)

Jan 2017
249 hikes

Aug 2017
221 hikes

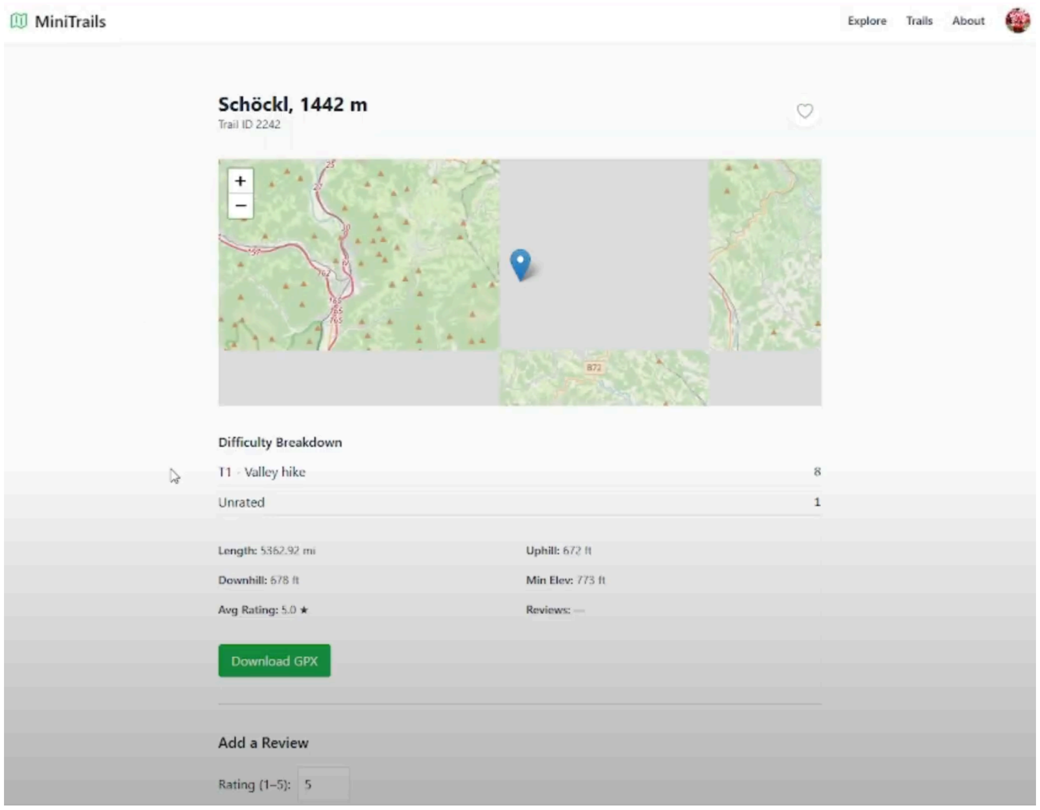
Jul 2017
206 hikes

Jun 2017
199 hikes

Oct 2017
186 hikes

```
/* Steepest Trails */
SELECT  TrailID,
       Name,
       ROUND(Uphill / NULLIF(Length3D,0), 3) AS GainPerMeter
FROM    dbo.Trail
WHERE   Length3D > 0
ORDER BY GainPerMeter DESC;
GO

/* Top Months */
SELECT  FORMAT(HI.StartTime, 'yyyy-MM') AS YearMonth,
       COUNT(*)                  AS NumHikes
FROM    dbo.Hike HI
WHERE   HI.StartTime >= DATEADD(MONTH, -12, SYSUTCDATETIME())
GROUP BY FORMAT(HI.StartTime, 'yyyy-MM')
ORDER BY COUNT(*) DESC;
GO
```

On trails page you can click on any trail to open up the details for that specific trail. This will show a map with the location of the trail along with information about it such as length, uphill, downhill, min elevation, and average rating. You can also enter a review which involves giving it a rating of 1-5, assigning it a difficulty of easy, medium, or hard, and including free text comments. When the user submits the review it gets saved to the database in the Reviews table.



Schöckl, 1442 m
Trail ID 2242

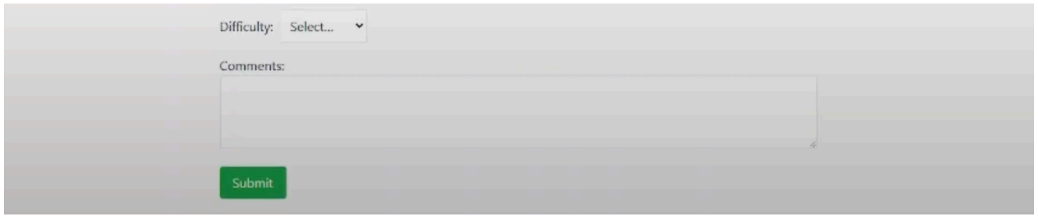
Difficulty Breakdown

T1 · Valley hike	8
Unrated	1

Length: 5362.92 mi	Uphill: 672 ft
Downhill: 678 ft	Min Elev: 773 ft
Avg Rating: 5.0 ★	Reviews: —

Add a Review

Rating (1–5): 5



Difficulty: Select...

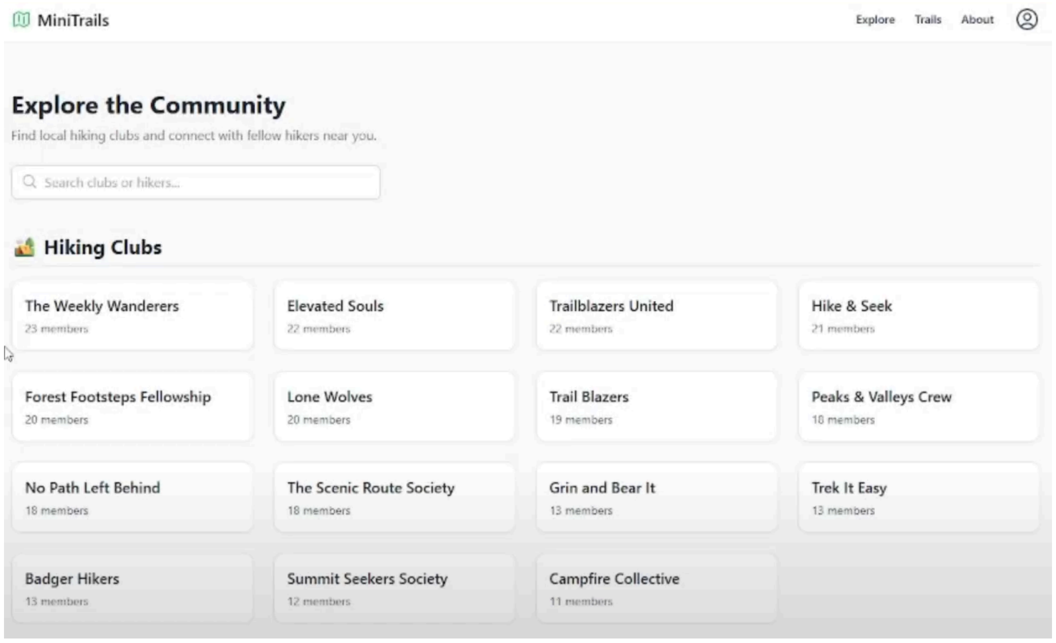
Comments:

```
    /* dbo.GetTrailSummary: aggregated stats for a trail */
    CREATE OR ALTER PROC dbo.GetTrailSummary
```

```
   @TrailID BIGINT
AS
BEGIN
   SET NOCOUNT ON;

   SELECT  T.TrailID,
          T.Name,
          AVG(R.Rating)              AS AvgRating,
          COUNT(R.ReviewID)          AS NumReviews,
          COUNT(HI.HikeID)           AS NumHikes,
          MIN(T.MinElevation)        AS MinElevation,
          MAX(T.MinElevation+T.Uphill)AS MaxElevation
   FROM    dbo.Trail  T
   LEFT JOIN dbo.Review R ON R.TrailID = T.TrailID
   LEFT JOIN dbo.Hike   HI ON HI.TrailID = T.TrailID
   WHERE   T.TrailID = @TrailID
   GROUP BY T.TrailID, T.Name;
END;
GO

/* dbo.AddReview: insert or update a review */
CREATE OR ALTER PROC dbo.AddReview
   @UserID     BIGINT,
   @TrailID    BIGINT,
   @Rating     INT,
   @Difficulty VARCHAR(255) = NULL,
   @Comments   VARCHAR(4000) = NULL
AS
BEGIN
   SET NOCOUNT ON;

   MERGE dbo.Review AS T
   USING (SELECT @UserID AS UserID, @TrailID AS TrailID) AS S
     ON T.UserID = S.UserID AND T.TrailID = S.TrailID
   WHEN MATCHED THEN
      UPDATE SET Rating     = @Rating,
                 Difficulty = @Difficulty,
                 Comments   = @Comments
   WHEN NOT MATCHED THEN
      INSERT (Comments, Difficulty, Rating, UserID, TrailID)
      VALUES (@Comments, @Difficulty, @Rating, @UserID, @TrailID);
END;
```

```
GO
```

The explore page shows off stats related to hiking clubs that are in our database. It lists all the clubs with the ability to search for clubs by name and for hikers by name to find the clubs that they are a part of.



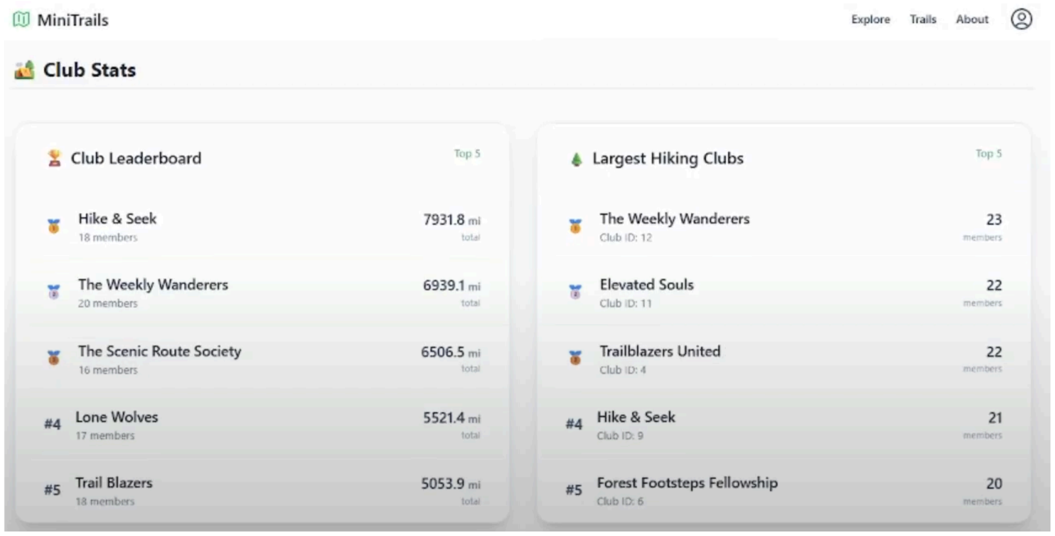
Explore the Community

Find local hiking clubs and connect with fellow hikers near you.

Hiking Clubs

The Weekly Wanderers 23 members	Elevated Souls 22 members	Trailblazers United 22 members	Hike & Seek 21 members
Forest Footsteps Fellowship 20 members	Lone Wolves 20 members	Trail Blazers 19 members	Peaks & Valleys Crew 18 members
No Path Left Behind 18 members	The Scenic Route Society 18 members	Grin and Bear It 13 members	Trek It Easy 13 members
Badger Hikers 13 members	Summit Seekers Society 12 members	Campfire Collective 11 members	

Below that are two tables: the club leaderboard which shows which club has logged the most hiking miles and largest hiking clubs which shows which clubs have the most members.



Club Stats

Club Leaderboard — Top 5

	Hike & Seek 18 members	7931.8 mi total
	The Weekly Wanderers 20 members	6939.1 mi total
	The Scenic Route Society 16 members	6506.5 mi total
#4	Lone Wolves 17 members	5521.4 mi total
#5	Trail Blazers 18 members	5053.9 mi total

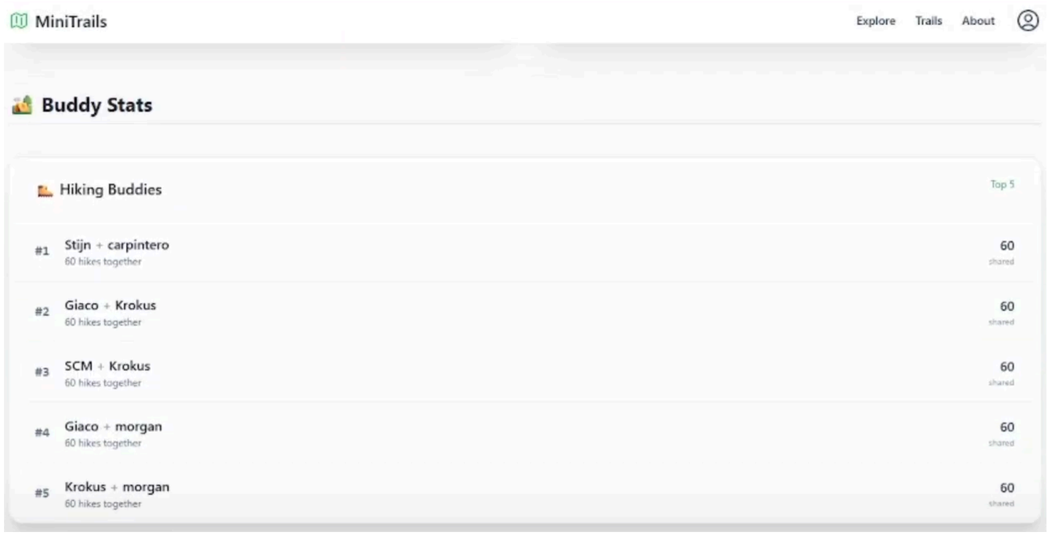
Largest Hiking Clubs — Top 5

	The Weekly Wanderers Club ID: 12	23 members
	Elevated Souls Club ID: 11	22 members
	Trailblazers United Club ID: 4	22 members
#4	Hike & Seek Club ID: 9	21 members
#5	Forest Footsteps Fellowship Club ID: 6	20 members

```
/* dbo.TopClubsByActivity: top clubs by hikes in a date range */
CREATE OR ALTER PROC dbo.TopClubsByActivity
```

```sql
    @StartDate DATETIME2,
    @EndDate   DATETIME2
AS
BEGIN
    SET NOCOUNT ON;

    SELECT TOP 10
          HC.ClubID,
          HC.Name,
          COUNT(DISTINCT HI.HikeID) AS TotalHikes
    FROM   dbo.HikingClub HC
    JOIN   dbo.Member     M  ON M.ClubID = HC.ClubID
    JOIN   dbo.Hiker      H  ON H.UserID = M.UserID
    JOIN   dbo.Hike       HI ON HI.HikeID = H.HikeID
    WHERE  HI.StartTime BETWEEN @StartDate AND @EndDate
    GROUP  BY HC.ClubID, HC.Name
    ORDER  BY TotalHikes DESC;
END;
GO

/* Club membership counts (L2) */
SELECT  C.ClubID,
        C.Name,
        COUNT(*) AS NumMembers
FROM    dbo.HikingClub C
JOIN    dbo.Member     M ON M.ClubID = C.ClubID
GROUP BY C.ClubID, C.Name
ORDER BY NumMembers DESC;
GO
```

Next is a table called buddy stats which shows the hiking pairs that have hiked the most miles together, which queries the database looking for hikes with multiple hikers attached.


MiniTrails

Explore Trails About

Buddy Stats

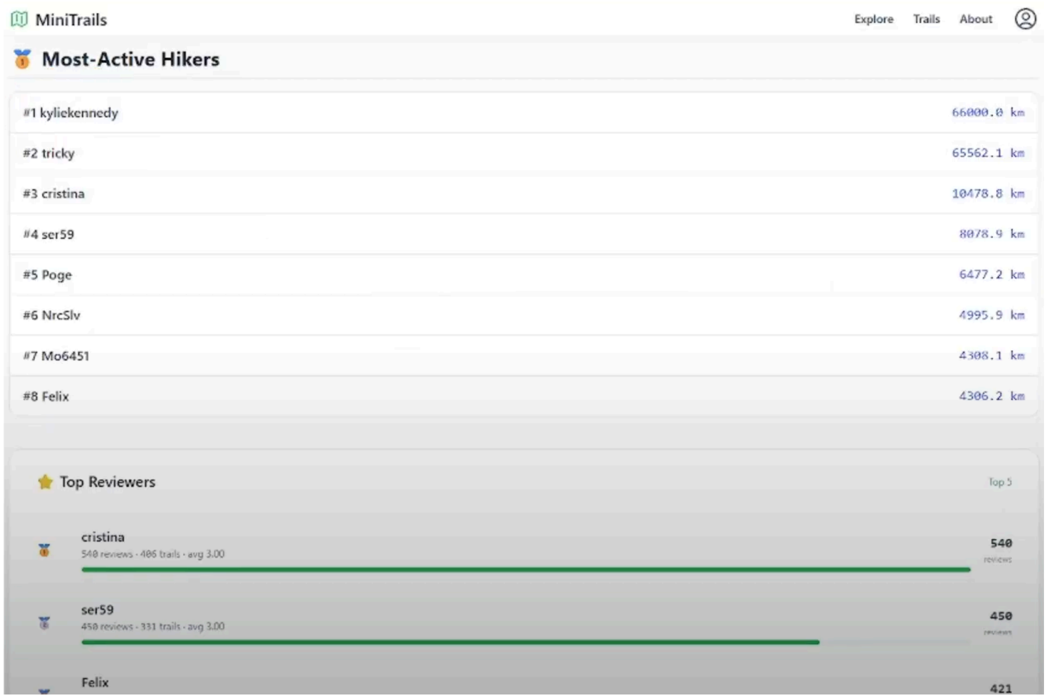
Hiking Buddies

Top 5

#1	Stijn + carpintero 60 hikes together	60 shared
#2	Giaco + Krokus 60 hikes together	60 shared
#3	SCM + Krokus 60 hikes together	60 shared
#4	Giaco + morgan 60 hikes together	60 shared
#5	Krokus + morgan 60 hikes together	60 shared

```
/* Buddy Stats - Pairs of users who hiked together */
SELECT  U1.Name AS HikerA,
        U2.Name AS HikerB,
        COUNT(*) AS SharedHikes
FROM    dbo.Hiker H1
JOIN    dbo.Hiker H2
        ON H1.HikeID = H2.HikeID
       AND H1.UserID < H2.UserID
JOIN    dbo.Users U1 ON U1.UserID = H1.UserID
JOIN    dbo.Users U2 ON U2.UserID = H2.UserID
JOIN    dbo.Hike  HI ON HI.HikeID = H1.HikeID
WHERE   DATEDIFF(MINUTE, HI.StartTime, HI.EndTime) > 0
GROUP BY U1.Name, U2.Name
ORDER BY COUNT(*) DESC;
GO
```

Lastly, there are components for the most active hikers and reviews which show the hikers with the most kilometers logged and most reviews submitted respectively.



Most-Active Hikers

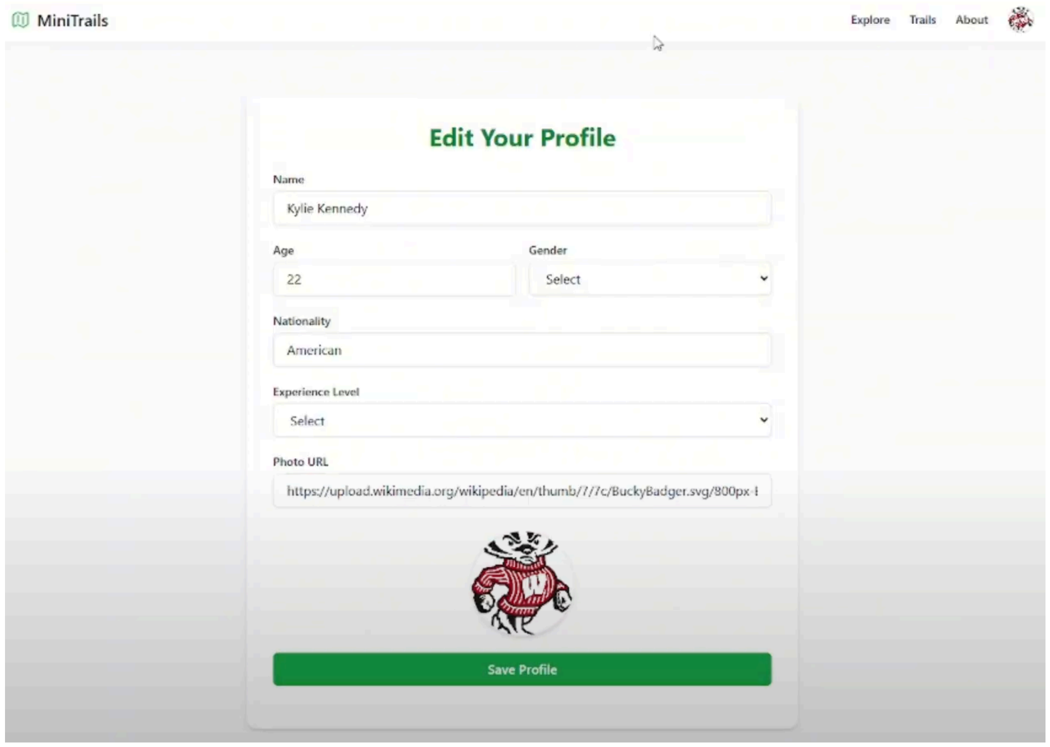
```
#1 kyliekennedy     66000.0 km
#2 tricky           65562.1 km
#3 cristina         10478.8 km
#4 ser59             8078.9 km
#5 Poge              6477.2 km
#6 NrcSlv            4995.9 km
#7 Mo6451            4308.1 km
#8 Felix             4306.2 km
```

Top Reviewers   Top 5

```
cristina                                        540
540 reviews · 406 trails · avg 3.00             reviews
ser59                                           450
450 reviews · 331 trails · avg 3.00             reviews
Felix                                           421
```

```
/* Most-active hikers and their total 3-D distance */
SELECT  U.UserID,
       U.Name,
       SUM(T.Length3D) AS TotalDistance_m
FROM    dbo.Hiker H
JOIN    dbo.Users U  ON U.UserID  = H.UserID
JOIN    dbo.Hike  HI ON HI.HikeID = H.HikeID
JOIN    dbo.Trail T  ON T.TrailID = HI.TrailID
GROUP BY U.UserID, U.Name
HAVING   SUM(T.Length3D) > 50000      -- >50 km
ORDER BY TotalDistance_m DESC;
GO

/* Top Reviewers */
SELECT  U.UserID,
       MAX(U.Name) AS Name,
       COUNT(*) AS ReviewCount
FROM    dbo.Users  U
INNER  JOIN dbo.Review R ON R.UserID = U.UserID
GROUP BY U.UserID
ORDER BY COUNT(*) DESC;
GO
```

The last page is a profile page where users can create or update their user account. It has fields for their name, age, gender, nationality, experience level, and photo. This all gets saved into the database into our Users table which is used in queries to return information specific to the user and larger demographic based queries like what's on the Explore page.



Name: Kylie Kennedy
Age: 22
Gender: Select
Nationality: American
Experience Level: Select
Photo URL: https://upload.wikimedia.org/wikipedia/en/thumb/7/7c/BuckyBadger.svg/800px-I

```
CREATE   PROCEDURE dbo.CreateUserAccount
   @Name     NVARCHAR(255),
   @Email    NVARCHAR(255),
   @Password NVARCHAR(255)   -- assume this is hashed already
AS
BEGIN
   SET NOCOUNT ON;

   -- 1. Check for duplicate email
   IF EXISTS (SELECT 1 FROM dbo.Users WHERE Email = @Email)
   BEGIN
      RAISERROR('Email already in use.', 16, 1);
      RETURN;
   END

   -- 2. Insert the user
```

```
  INSERT INTO dbo.Users (Name, Email, Password)
  VALUES (@Name, @Email, @Password);
END;

CREATE   PROCEDURE UpdateUserProfile
 @UserID INT,
 @PhotoURL VARCHAR(1000),
 @Name NVARCHAR(100),
 @ExperienceLevel NVARCHAR(50),
 @Gender NVARCHAR(20),
 @Nationality NVARCHAR(50),
 @Age INT
AS
BEGIN
 UPDATE Users
 SET
  PhotoURL = @PhotoURL,
  Name = @Name,
  Gender = @Gender,
  Nationality = @Nationality,
  Age = @Age
 WHERE UserID = @UserID;
END;
```

- Evaluation/Testing PDF
- 🎬 MiniTrails Demo.mp4

III. Conclusion

We learned how to work collaboratively to create a web app using Microsoft SQL, React, and Node JS. Points we found interesting were how we can create aspects that are interactive and engaging with users(such as storing data like profile photos and having community-building leaderboards) as well as having site sections update in real time using SQL. It was interesting to learn how to write code in a non-SQL language that called SQL code to interact with the database.

We each found different parts of development more or less interesting. For example, Kylie found the process and logic behind queries to be less invigorating than the outcome on the frontend. On the other hand, Alex and Josiah found that aspect to be more mentally stimulating. Most importantly, we collectively learned how to work as a team which is a valuable skill in development.

IV. References

We learned how to use the following tools in developing MiniTrails:

- Tanstack Query
- Using Vite
- Using SSMS
- Using Tailwind

They each offered flexibility either in connecting to the DB or to interface styling.