

Trailz

Software Design Document

CS 364 Section 2

Instructor: Stanley Wilde

July 18th, 2020

Stake Holders

- Ryan Arveseth
- Alex Baker
- Absias Baldiviezo Aguilar
- Katelyn Borne
- Rachel Bowery
- Chandler Childs
- Shannon Day
- Paul Decker
- Jonathan Garcia
- Weston Gavin
- Adam Gerhartz
- Eric Guerrero
- Herman Kravchenko
- Lucia Mata
- Josh Mayberry
- Kyle McGhie
- Gage Mikos
- Irina O'Hara
- Paul Porter
- Casey Spence
- Jonathan Steele

Version History

Date	Version	Notes
6/13/2020	0	<ul style="list-style-type: none">• Created Traceability Matrix as well as Entity table
6/20/2020	1	<ul style="list-style-type: none">• Added formatting, described 4 entities
7/4/2020	2	<ul style="list-style-type: none">• Finalize formatting• Added 41 entities• Added ERD design elements
7/11/2020	3	<ul style="list-style-type: none">• Added ERD and design elements.• Flowcharts and Pseudocode• Design Concerns• Appendix
7/18/2020	Final	<ul style="list-style-type: none">• Reworked the whole document

Table Of Contents

1 Introduction	4
1.1 Purpose	4
1.2 Scope.....	4
1.3 Glossary.....	5
2 Data Model	8
2.1 ERD	8
2.2 Design Elements.....	9
2.2.1 Table: Additional_Info	9
2.2.2 Table: Amenity.....	9
2.2.3 Table: Amenity_Trail	9
2.2.4 Table: Comment	10
2.2.5 Table: Comment_Thread	10
2.2.6 Table: Friend	10
2.2.7 Table: Friend_Request.....	10
2.2.8 Table: Hike	11
2.2.9 Table: Hike_Route_Point.....	11
2.2.10 Table: Hiking_Equipment	12
2.2.11 Table: Hiking_Equipment_Trail	12
2.2.12 Table: Location	12
2.2.13 Table: Message	12
2.2.14 Table: Message_Thread.....	13
2.2.15 Table: Message_Thread_Friend.....	13
2.2.16 Table: Photo.....	13
2.2.17 Table: Photo_Trail.....	13
2.2.18 Table: Plant.....	14
2.2.19 Table: Plant_Trail	14
2.2.20 Table: Point_Of_Interest	14
2.2.21 Table: Point_Of_Interest_Trail	15
2.2.22 Table: Ranger	15
2.2.23 Table: Review.....	15
2.2.24 Table: Route_Point	16
2.2.25 Table: Scavenger_Hunt.....	16
2.2.26 Table: Scavenger_Hunt_POI.....	16
2.2.27 Table: Settings.....	16
2.2.28 Table: Trail	17
2.2.29 Table: Trail_Route_Point.....	17
2.2.30 Table: User.....	18
2.2.31 Table: User_Trail	18
2.2.32 Table: Wildlife.....	19
3 Design Views	20
3.1 Data Management.....	20
3.1.1 AccessFirebase.....	20
3.1.2 AdditionalInfoDM.....	24
3.1.3 AmenityDM.....	26
3.1.4 CommentThreadDM.....	28
3.1.5 FriendRequestDM	32
3.1.6 HikeDM	35
3.1.7 HikingEquipmentDM	41
3.1.8 LocationDM.....	43

3.1.9 MessageThreadDM	48
3.1.10 Photo	53
3.1.11 PhotoDM	56
3.1.12 PlantDM.....	60
3.1.13 PointOfInterestDM.....	61
3.1.14 RangerDM	65
3.1.15 ReviewDM.....	67
3.1.16 ScavengerHuntDM.....	71
3.1.18 SettingsDM.....	74
3.1.19 TrailDM	76
3.1.20 UserDM.....	81
3.1.21 WildlifeDM.....	88
3.2 Home.....	90
3.2.1 HomeController.....	90
3.2.2 SettingsController.....	93
3.3 Login and Registration	97
3.3.1 ForgotMyPasswordController.....	97
3.3.2 LoginController	100
3.3.3 RegisterController	102
3.3.4 TwoFactorAuthentication	105
3.3.5 User.....	108
3.4 Exercise	115
3.4.1 Hike	115
3.4.2 Stopwatch.....	120
3.5 Trail Information	123
3.5.1 AdditionalInfoController.....	123
3.5.2 Amenity	126
3.5.3 AmenitiesController.....	127
3.5.4 BrowseController	129
3.5.5 Comment.....	131
3.5.6 CommentThreadController.....	133
3.5.7 EmergencyContactInfoController.....	137
3.5.8 GeneralInfoController	139
3.5.9 HikingEquipment	142
3.5.10 Plant	143
3.5.11 PointOfInterest	144
3.5.12 PointsOfInterestController	147
3.5.13 Review.....	149
3.5.14 ReviewController.....	150
3.5.15 SafetyGuidelinesController.....	154
3.5.16 Trail	157
3.5.17 TrailDescriptionController	161
3.5.18 TrailSuggestionGenerator.....	167
3.5.19 TraversalInfoController	170
3.5.20 Weather.....	172
3.5.21 Wildlife.....	173
3.6 Social	175
3.6.1 EditProfileController.....	175
3.6.2 FavoriteTrailsController	177
3.6.3 FriendsController	178
3.6.4 FriendRequest.....	181
3.6.5 FriendRequestController	182
3.6.6 Location.....	185

3.6.7 LocationTracker	188
3.6.8 Message.....	189
3.6.9 MessageThreadController.....	191
3.6.10 MessageThreadsController	196
3.6.11 ScavengerHuntController	199
3.6.12 UserProfileController	202
3.7 Map	204
3.7.1 AllTrailsController	204
3.7.2 FilterController	207
3.7.3 Icon	210
3.7.4 Map	212
3.7.5 MapLegend.....	215
4 Appendix.....	217

1 Introduction

1.1 Purpose

Trailz, referred to in the document as “The app”, is a mobile device application that allows individuals to share, view and rate various hiking trails using the app. The app improves the hiking and preparation for a given hiking trail by allowing individuals to share their experiences and recommendations regarding said trail, providing a trail description, a trail map, safety guidelines and points of interest. map, safety guidelines and points of interest.

1.2 Scope

The app will be for users who are new to hiking or are looking for hiking trails to use. There are several individuals who are looking for help in finding information on trails, such as which trails to hike on, what to bring, what to be cautious of, and different points of interest along the trail. There are also several individuals that are willing to share the information they have about these topics and questions. This app seeks to provide a platform to spread and share the information by having all the needed information about hiking in a single consolidated mobile application.

1.3 Glossary

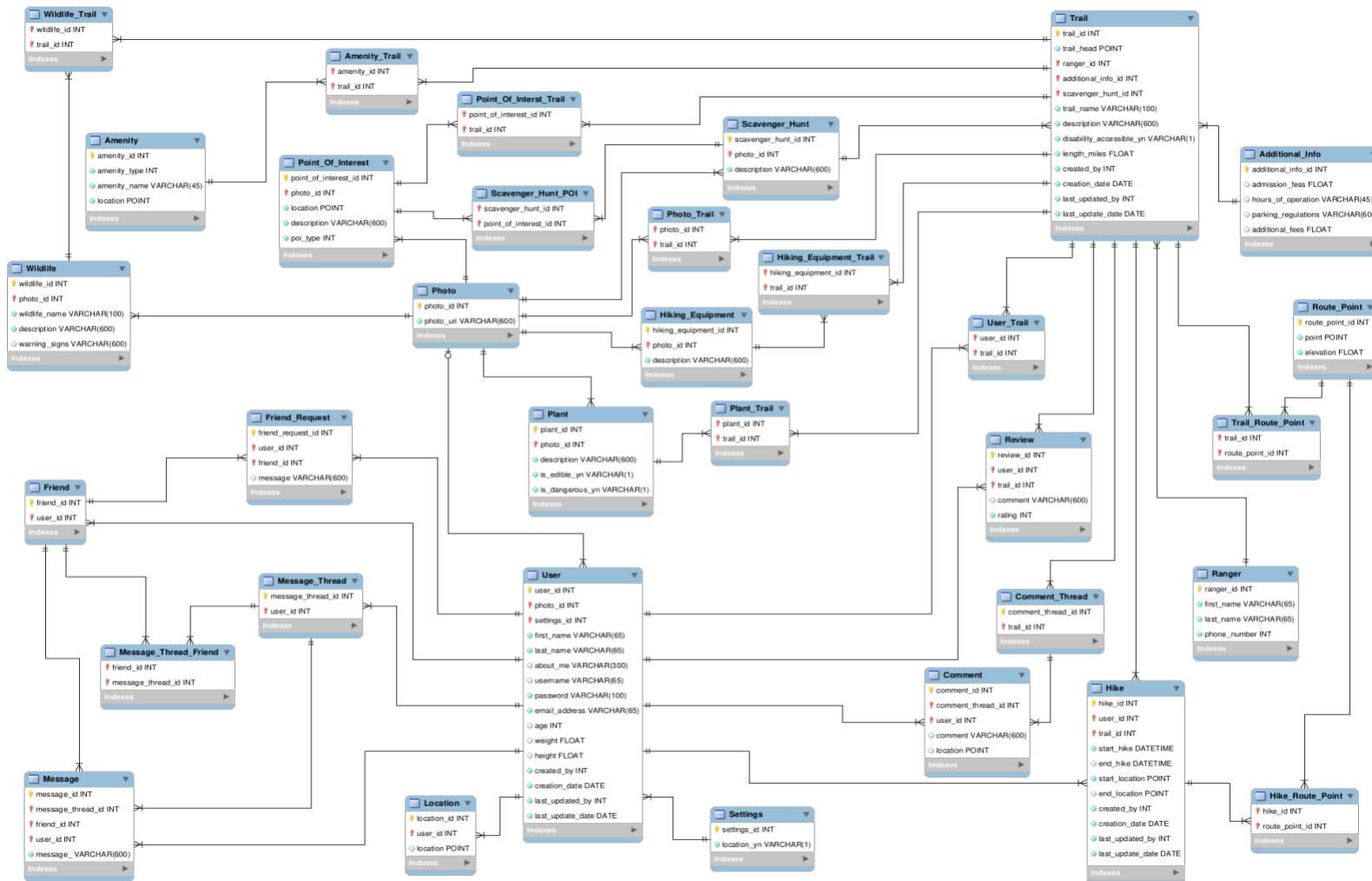
Term	Definition
Admissions Fees	The price paid for the right or permission to enter a Trail.
Amenities	<p>Amenities is defined as:</p> <ul style="list-style-type: none"> • Restrooms • Trailhead/Map • Water fountains • Fire pits • Trail markers • Information centers • Trash cans • Campsites • Picnic tables • Illuminated Trails
Application	A program that Users may utilize to perform specific tasks.
Bottom Navigation Panel	A section of graphical User interface which aids the User to accessing the Application's features, including: the app's home page, settings, hikes, as well as all the other app pages.
Card	A group of information within a small area, with a border – almost like a post-Card on the screen. This information may include a background image, hike details, ratings, and fitness tracking.
Cloud Storage	A service provided by large internet consumer companies where data is stored in globally accessible locations called Buckets (not databases).
Comment	A block of text information provided by a User related to a Trail Informational Subsection.
Comment Rating System	A System where Users may vote for or against specific Comments or report them for inappropriate content. Comments that are highly rated are displayed first.
Comment System	A persistent system that Users may provide Comments attached to some aspect of the Trailz app, be it general Comments on a Trail or a specific informational subsection. The Trailz app will save and display Comments in their appropriate locations. The Comment System is to be paired with the Comment Rating System.
Cross Platform Compatible	Specifies that your Application/website works on different platforms/devices. Examples are desktop computers, laptops, tablets, smartphones, etc.
Current Location	The current geographical area that the User is using the app from.
Difficulty	A Trail that requires certain amount of strength, stamina, or a level of commitment that is determined by a numerical value between 1 and 10, 10 being the hardest Difficulty.
Disability Accessible	Public accommodation for people of any age or physical condition to participate in the Trail.

Edible Plant	A plant suitable by nature to be fit or safe to eat.
Elevation	The height above a given level, especially sea level.
Event Host	The User who creates a planned social occasion in the Application. They can edit and share the event location, time and date.
Fab Button	Floating action button – a button which performs the most common action on a screen. Typically, a Fab Button is displayed as a circular shape with an icon in its center.
Fee	To make a payment to someone in return for services.
Grade	The incline of the Trail path.
Hazards	Threats to personal safety.
Hiking Equipment	The equipment taken on outdoor trips according to the duration, distance, and environment. Hikers may take with them equipment ranging from bug spray, water, hiking boots, sunscreen, and hat.
Hunting Location	A place or area used for hunting.
Image Card-Button	A button which does not appear the same as a standard button. Image Card-Buttons appear as images, with or without text inside of them, which may be clicked like a button. Image Card-Buttons are usually larger than regular buttons as well.
Instant Messaging	A type of online chat which offers real-time text transmission within the Application.
Irrelevant Warnings	Warnings which have previously impacted the Trail, but do not pertain to it anymore. Users should be aware of these warnings in case they ever become relevant again.
Long Pressing	When the User touches the screen and keeps their finger pressed down in a specific place for two or more seconds.
Map Window	The box which encompasses the map for the Trail. This window is small yet shows the full Trail to the User
Modal	An in-app popup which notifies the User of important information or necessary pending actions. Pending actions may include the User toggling location tracking within the app's settings.
Offline Usability	When there is no cell phone connection, data will be stored locally on the phone to be uploaded to the servers once a connection becomes available.
Parking Regulations	Parking lot rules or directions in which all vehicles must abide by.
Photos	A representation of a Trail or scene.
Point of Interest	A specific point location that someone may find useful or interesting.
Poisonous Plant	A different group from plants which cause illness if eaten in very large amounts or when touched can cause illness.

Predatory Wildlife	Animals that may potentially present a danger to the health or welfare of the people.
Profile Owner	A User who has signed up for the social aspect of a profile page.
Push Notification	An automated message sent by the Application to the User notifying them while the Application is not open.
Registration	The process of creating a new User in the Application
Relevant Warnings	Warnings which still pertain to the Trail. A relevant Warning means that the Warning is still active, and readers should be aware of it.
Safety Concerns	An important identified risk, important potential risk or important missing information.
Season	Any of the four divisions of the year (spring, summer, fall, and winter) marked by different weather patterns and daylight hours.
Terrain	The arrangement of the physical features of an area
Thread	The conversation history. A list of messages sent until that point in time.
Trail	A geographical path with a predefined start and finish that is in a natural setting of some kind.
Trailhead	The geographical starting point of a Trail. Often accompanied by a map or marker to indicate the Trail's location.
Trailz	The name given to the Application that this document describes.
Trail Information Subsections	Trail Information must have the following informational categories related to each Trail: <ul style="list-style-type: none"> • General Information • Emergency Contacts • Points of Interest • Suggested Items to Bring • Traversal • Safety Guidelines • Amenities
Two-Factor Authentication	Another layer of security which protects the User's account beyond only a Username and password. This usually includes an email or text which contains a verification code that is entered in the Application before the User has access to their personal information.
Warning	A Trail Warning is created by Users to caution other Users of obstacles, potential dangers, animals, weather, avalanches, etc.
Weather Forecast	A prediction of what the weather is likely to be for the next hour or few hours.
User	An individual who possesses the Application and desires to receive some benefit from its use.

2 Data Model

2.1 ERD



2.2 Design Elements

2.2.1 Table: Additional_Info

The **Additional_Info** table contains the following data: admission fees, hours of operation, parking regulations, and additional fees.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	additional_info_id	INT	NOT NULL	A surrogate key for unique identifier purposes
	admission_fees	FLOAT		An optional field consisting of admission fees
	hours_of_operation	VARCHAR(45)		A textual-numeric range that represents hours of operation
	parking_regulations	VARCHAR(600)		A brief textual description of any parking regulations
	additional_fees	FLOAT		An optional field consisting of additional fees

2.2.2 Table: Amenity

The **Amenity** table contains the following data: a type of amenity, an amenity name, and a location.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	amenity_id	INT	NOT NULL	A surrogate key for unique identifier purposes
	amenity_type	INT	NOT NULL	A type identifier to specify which type of amenity
	amenity_name	VARCHAR(45)	NOT NULL	A textual name of the amenity
	location	POINT	NOT NULL	A geographical location of the amenity

2.2.3 Table: Amenity_Trail

The **Amenity_Trail** table is a junction table between the Trail table and the Amenity table. A trail may contain many amenities. A specific amenity can be included in many trails. The Amenity_Trail table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	amenity_id	INT	NOT NULL	A non-unique reference to the Amenity table
Primary (Composite), Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table

2.2.4 Table: Comment

The **Comment** table contains the following data: a reference to a comment thread, a reference to a user, a comment, and a location on the map.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	comment_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	comment_thread_id	INT	NOT NULL	A non-unique reference to the Comment_Thread table
Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table
	comment	VARCHAR(600)		The textual comment written by the user
	location	POINT		A plotted point on the map

2.2.5 Table: Comment_Thread

The **Comment_Thread** table contains the following data: a reference to a trail.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	comment_thread_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	trail_id	POINT	NOT NULL	A non-unique reference to the Trail table

2.2.6 Table: Friend

The **Friend** table contains the following data: a reference to a user.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	friend_id	INT	NOT NULL	Surrogate key for unique identifier purposes
Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table

2.2.7 Table: Friend_Request

The **Friend_Request** table contains the following data: a reference to a user, a reference to a friend, and a message

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	friend_request_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table
Foreign	friend_id	INT	NOT NULL	A non-unique reference to the Friend table
	message	VARCHAR(600)		An optional, textual message a user may send to another user.

2.2.8 Table: Hike

The **Hike** table contains the following data: a reference to a user, a reference to a trail, a starting time, an ending time, a starting location, and an ending location.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	hike_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table
Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table
	start_hike	DATETIME	NOT NULL	A time when the user started the hike
	end_hike	DATETIME		A time when the user ended the hike
	start_location	POINT	NOT NULL	A geographical location of where the user started the hike
	end_location	POINT		A geographical location of where the user ended the hike
	created_by	INT	NOT NULL	For DBA usage, specifying who created the row
	creation_date	DATE	NOT NUL	For DBA usage, specifying when the row was created
	last_updated_by	INT	NOT NULL	For DBA usage, specifying who manipulated the row
	last_update_date	DATE	NOT NULL	For DBA usage, specifying when the row was manipulated

2.2.9 Table: Hike_Route_Point

The **Hike_Route_Point** table is a junction table between the Hike Table and the Route_Point table. The hike entity may have many route points. A specific route point may have many hikes if the user has walked the same paths. The Hike_Route_Point table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	hike_id	INT	NOT NULL	A non-unique reference to the Hike table
Primary (Composite), Foreign	route_point_id	INT	NOT NULL	A non-unique reference to the Route_Point table

2.2.10 Table: Hiking_Equipment

The **Hiking_Equipment** table contains the following data: a reference to a photo and a description.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	hiking_equipment_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	photo_id	INT	NOT NULL	A non-unique reference to the Photo table
	description	VARCHAR(600)	NOT NULL	A brief textual description of the hiking equipment

2.2.11 Table: Hiking_Equipment_Trail

The **Hiking_Equipment_Trail** table is a junction table between the Trail table and the Hiking_Equipment table. The trail entity may have many hiking equipment suggestions. A specific hiking equipment suggestion may find itself valid for many trails. The Hiking_Equipment_Trail table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	hiking_equipment_id	INT	NOT NULL	A non-unique reference to the Hiking_Equipment table
Primary (Composite), Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table

2.2.12 Table: Location

The **Location** table contains the following data: a reference to a user and a location.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	location_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table
	location	POINT		A geographic location of the user

2.2.13 Table: Message

The **Message** table contains the following data: a reference to a message thread, a reference to a friend, a reference to a user, and a message.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	message_id	INT	NOT NULL	Surrogate key for unique identifier purposes
Foreign	message_thread_id	INT	NOT NULL	A non-unique reference to the Message_Thread table
Foreign	friend_id	INT		A non-unique reference to the Friend table
Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table
	message	VARCHAR(600)	NOT NULL	A simple textual message

2.2.14 Table: Message_Thread

The **Message_Thread** table contains the following data: a reference to a user.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	message_thread_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table

2.2.15 Table: Message_Thread_Friend

The **Message_Thread_Friend** table is a junction table between the Message_Thread table and the Friend table. The message thread entity may include many friends. A specific friend may find himself or herself on many message threads. The Message_Thread_Friend table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	friend_id	INT	NOT NULL	A non-unique reference to the Friend table
Primary (Composite), Foreign	message_thread_id	INT	NOT NULL	A non-unique reference to the Message_Thread table

2.2.16 Table: Photo

The **Photo** table contains the following data: a reference to a trail and a photo uri.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	photo_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table
	photo_uri	VARCHAR(600)	NOT NULL	A URI string that accesses the photo from the S3 bucket

2.2.17 Table: Photo_Trail

The **Photo_Trail** table is a junction table between the Trail table and the Photo table. The trail entity may have many photos. A specific photo may find itself on many trails. The Photo_Trail table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	photo_id	INT	NOT NULL	A non-unique reference to the Photo table
Primary (Composite), Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table

2.2.18 Table: Plant

The **Plant** table contains the following data: a reference to a photo, a description, an isEdible flag, and an isDangerous flag.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	plant_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	photo_id	INT	NOT NULL	A non-unique reference to the Photo table
	description	VARCHAR(600)	NOT NULL	A brief textual description of the plant
	is_edible_yn	VARCHAR(1)	NOT NULL	A flag stating whether the plant is edible or not
	is_dangerous_yn	VARCHAR(1)	NOT NULL	A flag stating whether the plant is dangerous or not

2.2.19 Table: Plant_Trail

The **Plant_Trail** table is a junction table between the Trail table and the Plant table. The trail entity may have many plants. A plant could be included of many trails. The Plant_Trail table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	plant_id	INT	NOT NULL	A non-unique reference to the Plant table
Primary (Composite), Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table

2.2.20 Table: Point_Of_Interest

The **Point_Of_Interest** table contains the following data: a reference to a photo, a location, a description, and a type of point of interest.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	point_of_interest_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	photo_id	INT	NOT NULL	A non-unique reference to the Photo table
	location	POINT	NOT NULL	A location or point on the map
	description	VARCHAR(600)	NOT NULL	A brief, textual description of the point of interest
	poi_type	INT	NOT NULL	A type identifier for points of interest

2.2.21 Table: Point_Of_Interest_Trail

The **Point_Of_Interest_trail** table is a junction table between the Trail table and the Point_Of_Interest table. The trail entity may have many points of interest. A specific point of interest may find itself on many overlapping trails. The Point_Of_Interest_Trail handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	point_of_interest_id	INT	NOT NULL	A non-unique reference to the Point_Of_Interest table
Primary (Composite), Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table

2.2.22 Table: Ranger

The **Ranger** table contains the following data: a first name, a last name, and a phone number.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	ranger_id	INT	NOT NULL	A surrogate key for unique identifier purposes
	first_name	VARCHAR(65)	NOT NULL	The ranger's first name
	last_name	VARCHAR(65)	NOT NULL	The ranger's last name
	phone_number	INT	NOT NULL	The ranger's phone number

2.2.23 Table: Review

The **Review** table contains the following data: a reference to a user, a reference to a trail, a comment, and a rating.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	review_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table
Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table
	comment	VARCHAR(600)	NOT NULL	A brief, textual comment
	rating	INT	NOT NULL	A numeric rating

2.2.24 Table: Route_Point

The **Route_Point** table contains the following data: a point and an elevation.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	route_point_id	INT	NOT NULL	A surrogate key for unique identifier purposes
	point	POINT	NOT NULL	A geographic location saved as a lat/long
	elevation	FLOAT	NOT NULL	The elevation (interpreted in miles) of the point

2.2.25 Table: Scavenger_Hunt

The **Scavenger_Hunt** table contains the following data: a reference to a photo and a description.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	scavenger_hunt_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	photo_id	INT	NOT NULL	A non-unique reference to the Photo table
	description	VARCHAR(600)	NOT NULL	A simple textual description of the scavenger hunt rules and tasks

2.2.26 Table: Scavenger_Hunt_POI

The **Scavenger_Hunt_POI** table is a junction table between the Point_Of_Interest table and the Scavenger_Hunt table. The point of interest entity may have many scavenger hunts. A specific scavenger hunt may have many points of interest. The Scavenger_Hunt_POI table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	scavenger_hunt_id	INT	NOT NULL	A non-unique reference to the Scavenger_Hunt table
Primary (Composite), Foreign	point_of_interest_id	INT	NOT NULL	A non-unique reference to the Point_Of_Interest table

2.2.27 Table: Settings

The **Settings** table contains the following data: a location sharing flag

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	settings_id	INT	NOT NULL	A surrogate key for unique identifier purposes
	location_yn	VARCHAR(1)	NOT NULL	A flag for location sharing

2.2.28 Table: Trail

The **Trail** table contains the following data specific to a trail: a trail-head location, a reference to a ranger, a reference to additional information, a reference to a scavenger hunt, a trail-name, a description, a disability-accessible flag/boolean, and the length of the trail in miles.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	trail_id	INT	NOT NULL	A surrogate key for unique identifier purposes
	trail_head	POINT	NOT NULL	A non-unique, geographic location of the trail's head
Foreign	ranger_id	INT	NOT NULL	A non-unique reference to the Ranger table
Foreign	additional_info_id	INT	NOT NULL	A non-unique reference to the Additional_Info table
Foreign	scavenger_hunt_id	INT	NOT NULL	A non-unique reference to the Scavenger_Hunt table
	trail_name	VARCHAR(100)	NOT NULL	The textual name of the trail
	description	VARCHAR(600)	NOT NULL	The textual description of the trail
	disability_accessible_yn	VARCHAR(1)	NOT NULL	A flag saying if the trail is disability accessible
	length_miles	FLOAT	NOT NULL	The length of the hike (to be interpreted in miles)
	created_by	INT	NOT NULL	For DBA usage, specifying who created the row
	creation_date	DATE	NOT NULL	For DBA usage, specifying when the row was created
	last_updated_by	INT	NOT NULL	For DBA usage, specifying who manipulated the row
	last_update_date	DATE	NOT NULL	For DBA usage, specifying when the row was manipulated

2.2.29 Table: Trail_Route_Point

The **Trail_Route_Point** table is a junction table between the Trail table and the Route_Point table. The trail entity may have many route points. A specific route point may have many trails (overlapping trails). The Trail_Route_Point table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table
Primary (Composite), Foreign	route_point_id	INT	NOT NULL	A non-unique reference to the Route_Point table

2.2.30 Table: User

The **User** table contains the following data specific to the user: a reference to a photo, a reference to the user's settings, a first name, a last name, a biography, a username, a hashed password, an email address, an age, weight, and height.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary	user_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	photo_id	INT		A non-unique reference to the Photo table
Foreign	settings_id	INT	NOT NULL	A non-unique reference to the Settings table
	first_name	VARCHAR(65)	NOT NULL	A user-inputted first name
	last_name	VARCHAR(65)	NOT NULL	A user-inputted last name
	about_me	VARCHAR(300)		A non-unique, user-inputted textual biography
	username	VARCHAR(65)	NOT NULL	A unique system-generated username that is the same as the email address
	password	VARCHAR(100)	NOT NULL	A non-unique, hash-generated password
	email_address	VARCHAR(65)	NOT NULL	A unique, user-inputted email address
	age	INT		A user-inputted age
	weight	FLOAT		A user-inputted weight
	height	FLOAT		A user-inputted height
	created_by	INT	NOT NULL	For DBA usage, specifying who created the row
	creation_date	DATE	NOT NULL	For DBA usage, specifying when was the row created
	last_updated_by	INT	NOT NULL	For DBA usage, specifying who manipulated the row
	last_update_date	DATE	NOT NULL	For DBA usage, specifying when the row was manipulated

2.2.31 Table: User_Trail

The **User_Trail** table is a junction table between the Trail table and the User table. The trail entity may have many users. A specific user may have many trails. The User_Trail table handles this many-to-many relationship.

KEY	FIELD NAME	TYPE	NULL?	DESCRIPTION
Primary (Composite), Foreign	user_id	INT	NOT NULL	A non-unique reference to the User table
Primary (Composite), Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table

2.2.32 Table: Wildlife

The **Wildlife** table contains the following data: a reference to a photo, a wildlife name, a description, and warning signs.

<i>KEY</i>	<i>FIELD NAME</i>	<i>TYPE</i>	<i>NULL?</i>	<i>DESCRIPTION</i>
Primary	wildlife_id	INT	NOT NULL	A surrogate key for unique identifier purposes
Foreign	photo_id	INT	NOT NULL	A non-unique reference to the Photo table
	wildlife_name	VARCHAR(100)	NOT NULL	A textual name of the wildlife
	description	VARCHAR(600)	NOT NULL	A brief textual description of the wildlife
	warning_signs	VARCHAR(600)		A brief textual description of the warning signs

Table: Wildlife_Trail

The **Wildlife_Trail** table is a junction table between the Trail table and the Wildlife table. The trail entity may have much wildlife. A specific wildlife may be included in many trails. The Wildlife_Trail table handles this many-to-many relationship.

<i>KEY</i>	<i>FIELD NAME</i>	<i>TYPE</i>	<i>NULL?</i>	<i>DESCRIPTION</i>
Primary (Composite), Foreign	wildlife_id	INT	NOT NULL	A non-unique reference to the Wildlife table
Primary (Composite), Foreign	trail_id	INT	NOT NULL	A non-unique reference to the Trail table

3 Design Views

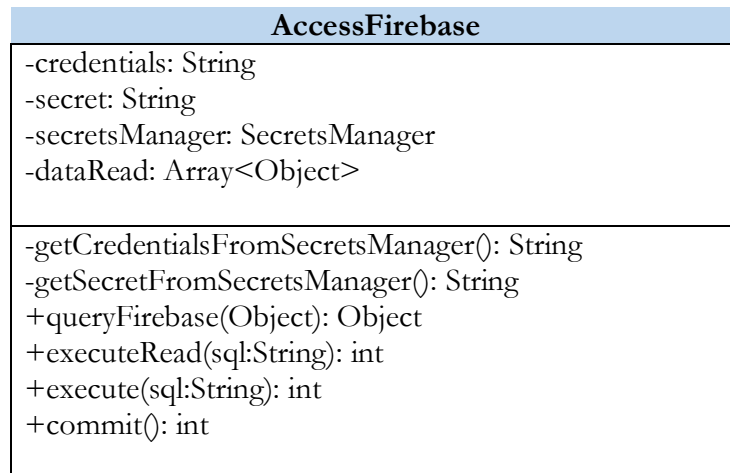
3.1 Data Management

This view maintains the data management classes of the application, enabling them to access the database, creating, reading, updating, and deleting with varying degrees of access.

3.1.1 AccessFirebase

Description: Provides access to the application Firebase database and file storage, subject to user authentication and security. Local device data is synced through this connection. The data for the individual users is kept using Cloud Storage. Users will only have access to their own data. It will also be where the data for the trails will be kept. All users will have access to the trail data. This class will get the secret and credentials from a SecretsManager and will allow all the different classes to access Firebase through its authorization.

Class Diagram



Design Elements

3.1.1.1.1 credentials: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the credentials that enables the app to communicate with Firebase.
Description	A String that can be used to access the proper Firebase account.

3.1.1.1.2 secret: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the secret that enables the app to communicate with Firebase.
Description	A String that can be used to access the proper Firebase account.

3.1.1.1.3 secretsManager: SecretsManager

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Accesses the Firebase Secrets Manager to retrieve the credentials to access the account.
Description	A SecretsManager API object that allows FirebaseAccess to get the secret and credentials.

3.1.1.1.4 dataRead: Array<Object>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Accesses the Firebase Secrets Manager to retrieve the credentials to access the account.
Description	A SecretsManager API object that allows FirebaseAccess to get the secret and credentials.

3.1.1.1.5 getCredentialsFromSecretsManager(): String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This method returns the credentials from the SecretsManager.
Description	Returns a String after the SecretsManager API successfully completes.
Parameters	None
Returns	A String that represents the account credentials.
Functions Called	None

3.1.1.1.6 getSecretFromSecretsManager(): String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This method returns the secret from SecretsManager.
Description	Returns a String after the SecretsManager API successfully completes.
Parameters	None
Returns	A String that represents the account secret.
Functions Called	None

3.1.1.1.7 queryFirebase(Object): Object

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Once a FirebaseAccess object has been established, other classes may make firebase queries using our FirebaseAccess class.
Description	Packages up a query to Firebase using the credentials and secret and waits until it receives a response, returning it to the caller.
Parameters	Any kind of object that Firebase API's could expect.
Returns	Any kind of object that Firebase API's could return.
Functions Called	None

3.1.1.1.8 executeRead(SQL:String): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Executes to the database using the parameter SQL String
Description	Executes the SQL statement to the database. Set the dataRead to the data that is retrieved upon execution.
Parameters	None
Returns	The return code from the SQL execution
Functions Called	commit()

3.1.1.1.9 execute(sql:String): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Executes the provided SQL command
Description	Executes the provided SQL command through the firebase API's, first caching them before they are committed.
Parameters	sql: String
Returns	The SQL return code
Functions Called	commit(): int

3.1.1.1.10 commit(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Commits the cached SQL commands
Description	Commits the cached SQL commands
Parameters	None
Returns	The int return code
Functions Called	Firebase.

Design Concerns

<i>SRS</i>	<i>Content</i>
3.4.1	Trail Photos load in less than 3 seconds of entering a Browse screen.
3.4.2	Trail Photos load in less than 3 seconds of entering a Trail Description screen.
3.4.3	User Comments and ratings submit in less than 3 seconds after the User's submission.
3.5.1	There is one production server-side NoSQL database and one production client-side database utilizing device local storage.
3.5.2	The server-side database returns and receives requests in JSON formatting.
3.5.3	Server-side information is stored and formatted as: <ul style="list-style-type: none">● User data:<ul style="list-style-type: none">○ Username/display name○ User's email● A unique identifier for the User● User Comments and ratings of Comments<ul style="list-style-type: none">○ Trail data of which each entry shall consist of:<ul style="list-style-type: none">○ A unique identifier for that Trail○ Trail name○ Trail description○ Trail location○ Images associated with the Trail○ Amenities associated with the Trail○ Points of interest associated with the Trail○ Location data of the Trail
3.5.4	Users can submit Comments, Comment ratings, and Trail ratings to the server-side database.
3.5.5	Users can editing permissions of their User data as well as Trail ratings, Comment ratings, and Comments submitted by them.
3.5.6	For an individual User, client-side local storage stores a String containing a unique identifier secured using a hash.
3.5.7	Client-side and server-side storage store:

<i>SRS</i>	<i>Content</i>
	<ul style="list-style-type: none"> • A list of Trails favorited by the User, with each list item containing the name of the Trail and its unique identifier. • User data containing the User's Username, email, and unique identifier.
3.5.8	Perform an auto-sync that will push data onto the server-side database.
3.5.9	Trigger an auto-sync upon loading the Application.
3.5.10	Trigger an auto-sync every 30 seconds given changes to the local storage are detected.
3.7.2.1	Google Firebase service handles database access and data syncing with more than a million Users.
3.7.2.2	Support offline usage using the Firebase client library; data remains accessible in local storage,
3.7.3.1	Login credentials and personal information are only accessible to the User.
3.7.3.2	Connection performed using a TCP/IP connection using SSL
3.7.3.3	A privacy policy assures Users of data privacy

3.1.2 AdditionalInfoDM

Provides access to the Additional_Info table.

AdditionalInfoDM
-dm: AccessFirebase -additionalInfoId: int -hoursOfOperation: String -admissionsFees: float -parkingRegulations: String -additionalFees: float -sqlString: String
+read(additionalInfoId: int): int

3.1.2.1 DesignElements

3.1.2.1.1 dm: AccessFirebase

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.2.1.2 additionalInfoId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The ID of the additional info from the Additional Info table
Description	The integer ID of the additional info from the Additional Info table

3.1.2.1.3 hoursOfOperation: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The hours of operation of this trail
Description	A string representing the hours of operation of this trail

3.1.2.1.4 admissionFees: float

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Represents the admission fees
Description	A float that represents the admission fees in US dollars

3.1.2.1.5 parkingRegulations: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A description of parking regulations, if any
Description	A string description of parking regulations, if any

3.1.2.1.6 additionalFees: float

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Any additional fees required to reach the trail
Description	A float representing any additional fees required to reach the trail in US dollars

3.1.2.1.7 sqlString: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.2.1.8 read(additionalInfoId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Used to read a fees, hours of operation, parking regulations, additional fees
Description	Accesses the AdditionalInfo table to retrieve the following data: <ul style="list-style-type: none">• hoursOfOperation: String• admissionsFees: float• parkingRegulations: String• additionalFees: float Generates SQL statement using the additionalInfoID parameter
Parameters	additionalInfoId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

3.1.3 AmenityDM

Description: Contains the methods that enable access to Amenities given a specific trailId.

AmenityDM
-dm: AccessFirebase -amenities: Array<Amenity> -trailId: int -sqlString: String
+read(additionalInfoId: int): int

3.1.3.1 Design Elements

3.1.3.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.3.1.2 amenities: Array<Amenity>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The array of retrieved amenities
Description	The array of Amenity objects retrieved and deserialized from the database

3.1.3.1.3 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The ID of the trail these amenities are connected to
Description	The integer ID of the trail these amenities are connected to

3.1.3.1.4 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.3.1.5 read(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read a list of amenities from the database.
Description	Accesses the Amenity_Trail and Amenity table to retrieve the following data: <ul style="list-style-type: none">• iconType• location• description and then store it into an Array of deserialized Amenity objects. Generates an SQL statement with the trailId.
Parameters	additionalInfoId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

3.1.4 CommentThreadDM

Description: CommentThreadDM provides access to the CommentThread and Comment tables in the database.

Class Diagram

CommentThreadDM
-dm: AccessFirebase -comments: Array<Comment> -commentThreadId: int -sqlString: string -trailId: int
+createComment(commentThreadId: int, comment: Comment): int +read(trailId: int): int +updateComment(commentId: int, comment: Comment) : int +deleteComment(commentId: int) : int -verify(comment: Comment): bool

3.1.4.1 Design Elements

3.1.4.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.4.1.2 comments: Array<Comment>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the retrieved Comment objects from the database.
Description	An array of Comment objects that are associated with this particular comment thread.

3.1.4.1.3 commentThreadId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Unique id for a comment thread
Description	An integer that uniquely identifies a comment thread in the table CommentThread.

3.1.4.1.4 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.4.1.5 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the trailId for the trail.
Description	An integer trail ID that will be the reference point for the junction of these two tables.

3.1.4.1.6 createComment(commentThreadId: int, comment: Comment): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to create a new comment within this comment thread.
Description	<p>Creates a new comment within this thread by supplying the parent commentThreadId and the comment to be added into it to be added to the Comment table.</p> <p>Creates the SQL statement for this command.</p> <p>Calls verify before calling the SQL statement, if verify returns true, proceeds with the execution, else this method fails.</p>
Parameters	commentThreadId: int, comment: Comment
Returns	The return code from FireBase execution
Functions Called	verify(comment: Comment) dm.execute(sqlString)

3.1.4.1.7 read(trailId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Used to read a comment thread data from the database
Description	<p>Retrieves the commentThread using the provided trailId from the Comment_Thread table. Retrieves:</p> <ul style="list-style-type: none">commentThreadId: int <p>Then retrieves an array of comments using this commentThreadId, deserializing the select statement as it is retrieved into Comment objects, selecting these fields:</p> <ul style="list-style-type: none">comment_id: intcomment_thread_id: intuser_id: intcomment: stringlocation: Point <p>Creates the SQL statement for this command.</p>
Parameters	trailId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

3.1.4.1.8 updateComment(commentId: int, comment: Comment) : int

ITEM	DESCRIPTION
Type	Function
Purpose	Make a change in a particular comment in the given commentThread.
Description	<p>If verify is successful on the provided comment, this method replaces the specified comment in the Comment table given its commentId with this data from the provided comment:</p> <ul style="list-style-type: none">comment.locationcomment.comment <p>This method generates the SQL statement for this command.</p>
Parameters	commentId: int, comment: Comment
Returns	The return code from FireBase execution
Functions Called	verify(comment: Comment) dm.execute(sqlString)

3.1.4.1.9 deleteComment(commentId: int) : int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Removes the association between the comment and the comment thread, effectively deleting it, but retaining the comment itself for our records.
Description	Removes the comment_thread_id from the specified comment at commentId in the Comment table, provided that the comment_thread_id matches this object's commentThreadId integer. If not, this method fails. Generates the SQL command to be able to perform this command.
Parameters	commentId: int
Returns	The return code from FireBase execution
Functions Called	dm.execute(sqlString)

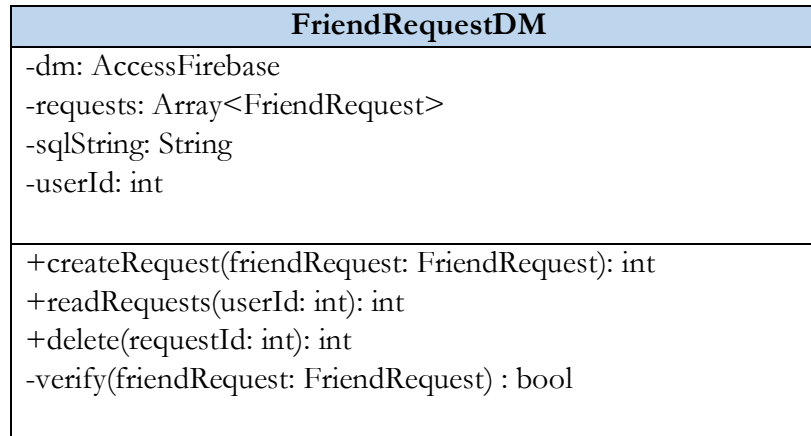
3.1.4.1.10 verify(comment: Comment): bool

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Verifies that the comment provided is appropriate and valid
Description	verifies that: comment.comment does not contain swear words comment.commentThreadId exists comment.userId exists
Parameters	comment: Comment
Returns	True if the comment passes the checks, false otherwise.
Functions Called	None

3.1.5 FriendRequestDM

Description: Provides access to Friend Requests table

Class Diagram



3.1.5.1 Design Elements

3.1.5.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.5.1.2 requests: Array<FriendRequest>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the retrieved Comment objects from the database.
Description	An array of Comment objects that are associated with this particular comment thread.

3.1.5.1.3 sqlString: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.5.1.4 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the userId for the user
Description	Unique integer user ID can be used to retrieve data from User table

3.1.5.1.5 createRequest(friendRequest: FriendRequest): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to create a new friend request
Description	<p>Creates a new friend request by supplying a friendRequest object to be added to the FriendRequest table</p> <p>Creates the SQL statement for this command.</p>
Parameters	friendRequest: FriendRequest
Returns	The return code from FireBase execution
Functions Called	dm.execute(sqlString)

3.1.5.1.6 readRequests(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read friend requests data from the database
Description	<p>Retrieves the friend request using the provided userId: int from the Friend Request table. Retrieves:</p> <ul style="list-style-type: none">• friendRequestId: int• userId: int• friendId: int• message: string <p>Creates the SQL statement for this command.</p>
Parameters	userId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

3.1.5.1.7 delete(requestId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Removes friendRequestId from the tabl
Description	Removes the friend request id from the table when user accepts a request. Before deleting a request it will use verify method to check that requestId exists in the table SQL command to be able to perform this command.
Parameters	requestId: int
Returns	The return code from FireBase execution
Functions Called	dm.execute(sqlString) verify(requestId): boolean

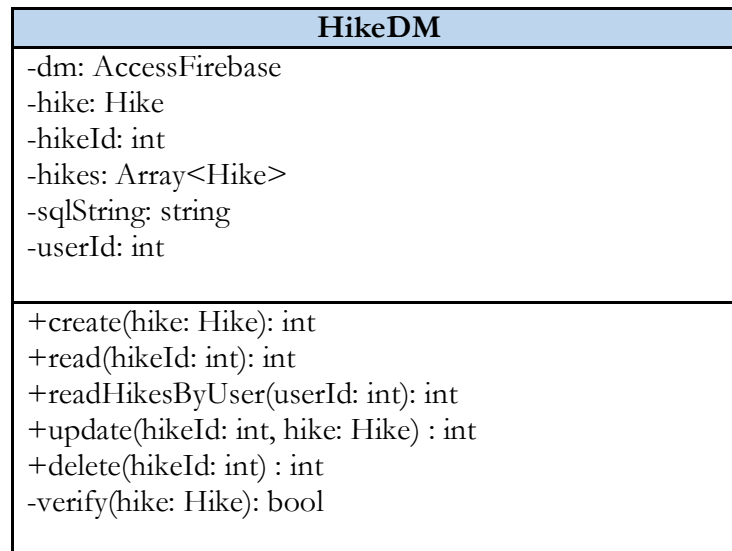
3.1.5.1.8 verify(friendRequestId: int) : bool

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Verifies that the comment provided is appropriate and valid
Description	verifies that: requestId exists
Parameters	friendRequestId: int
Returns	True if the comment passes the checks, false otherwise.
Functions Called	None

3.1.6 HikeDM

Provides access to the Hike, Hike_Route_Points, and Route_Points tables.

Class Diagram



3.1.6.1 Design Elements

3.1.6.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.6.1.2 hike: Hike

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the retrieved Hike object retrieved from the database.
Description	Contains the retrieved Hike object retrieved from the database.

3.1.6.1.3 hikeId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the hikeId for the hike.
Description	An integer hike ID that references this hike

3.1.6.1.4 hikes: Array<Hike>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains an array of Hike objects
Description	Contains an array of Hike objects for the readHikesByUser() method.

3.1.6.1.5 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.6.1.6 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the userId for the hike
Description	An integer user ID that exists for the method readHikesByUser()

3.1.6.1.7 create(hike: Hike): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to create a new hike in the Hike table
Description	<p>Creates a new hike by passing the hike object to the Hike table.</p> <p>Retrieves the primary key of this new Hike row and saves it to the hikeId variable.</p> <p>Also extracts the hike.routePointsArray array and creates rows in the Route_Points table that correspond to each Point in this array.</p> <p>Then creates rows in the Hike_Route_Points table that creates a junction between each Route_Points row that was created above to this hikeId.</p> <p>Creates the SQL statement for this command.</p> <p>Calls verify before calling the SQL statement, if verify returns true, proceeds with the execution, else this method fails.</p>
Parameters	hike: Hike
Returns	The return code from FireBase execution
Functions Called	verify(hike: Hike) dm.execute(sqlString)

3.1.6.1.8 read(hikeId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read hike data from the database
Description	<p>Retrieves the Hike using the provided hikeId from the Hike table. Retrieves:</p> <ul style="list-style-type: none">• hike_id: int• user_id: int• trail_id: int• end_hike: datetime• start_hike: datetime• start_location: Point• end_location: Point <p>Then deserializes this data into the hike object.</p> <p>Retrieves hike.route points by using the hikeId to make a select statement from a join between the Hike_Route_Points table and the Route_Points table, selecting everything that matches the hikeId, returning the array of Point objects into the hike.routePointsHike array.</p> <ul style="list-style-type: none">• location: Point <p>Creates the SQL statement for this command.</p>
Parameters	trailId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

3.1.6.1.9 readHikesByUser(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read hike data from the database
Description	<p>Retrieves an array of Hike objects using the provided userId from the Hike table. Retrieves:</p> <ul style="list-style-type: none">• hike_id: int• user_id: int• trail_id: int• end_hike: datetime• start_hike: datetime• start_location: Point• end_location: Point <p>Then deserializes this data into the hike object.</p> <p>Retrieves hike.route points by using the hikeId to make a select statement from a join between the Hike_Route_Points table and the Route_Points table, selecting everything that matches the hikeId, returning the array of Point objects into the hike.routePointsHike array.</p> <ul style="list-style-type: none">• location: Point <p>Creates the SQL statement for this command.</p>
Parameters	userId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

3.1.6.1.10 update(hikeId: int, hike: Hike) : int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Make a change in a particular hike, given the hikeId
Description	<p>If verify is successful on the provided hike, this method replaces the specified hike in the Hike table given its hikeId with this data from the provided comment:</p> <ul style="list-style-type: none"> • hike.endHike • hike.endLocation • hike.lengthHike • hike.startHike • hike.startLocation • hike.trailId • hike.userId <p>Will then update the associated Route_Points for every Point in hike.routePointsHike. This will not update preexisting Route_Points rows, but will store their primary keys in memory.</p> <p>Will then delete all rows from the Hike_Route_Points junction table that match the hikeId of this hike, and then re-establish the connection to Route_Points by adding rows to Hike_Route_Points for every Route_Point row in memory, coupling it to this hikeId.</p> <p>This method generates the SQL statement for this command.</p>
Parameters	hikeId: int, hike: Hike
Returns	The return code from FireBase execution
Functions Called	verify(hike: Hike) dm.execute(sqlString)

3.1.6.1.11 delete(hikeId: int) : int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Deletes the hike from the Hike table
Description	Deletes the row from the Hike table matching this hikeId. Then deletes all rows from the Trail_Route_Points table with this hikeId.
Parameters	hike: Hike
Returns	The return code from FireBase execution
Functions Called	dm.execute(sqlString)

3.1.6.1.12 verify(hike: Hike): bool

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Verifies that the comment provided is appropriate and valid
Description	verifies that: hike.endLocation is a real Point hike.startLocation is a real Point hike.lengthHike is non-negative hike.startHike is before hike.endHike hike.userId is this user's userId hike.trailId is a valid trailId hike.rotuePointsHike contains real Point objects
Parameters	hike: Hike
Returns	True if the comment passes the checks, false otherwise.
Functions Called	None

3.1.7 HikingEquipmentDM

Provides access to the Hiking_Equipment and Hiking_Equipment_Trail tables

HikingEquipmentDM
-dm: AccessFirebase -hikingEquipment: Array<HikingEquipment> -trailId: int -sqlString: String
+read(trailId: int): int

3.1.7.1 DesignElements

3.1.7.1.1 dm: AccessFirebase

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Enables access to Firebase.
Description	AccessFirebase object for direct database access.

3.1.7.1.2 hikingEquipment: Array<HikingEquipment>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains an array of objects for reference for classes utilizing this one.
Description	Contains an array of HikingEquipment objects for storage after deserialization and retrieval from the database.

3.1.7.1.3 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the trailId for the trail.
Description	An integer trail ID that will be the reference point for the junction of these two tables.

3.1.7.1.4 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.7.1.5 read(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read HikingEquipment objects from the database using the junction table and a trailId.
Description	<p>Accesses the Hiking_Equipment table to retrieve the following data utilizing a join with the Hiking_Equipment_Trail table and the trailId as a condition:</p> <ul style="list-style-type: none">• hiking_quipment_id: int• photo: Photo• description: string <p>Generates the SQL statement to facilitate this join.</p> <p>Deserializes the results into an appropriate number of HikingEquipment objects and stores them in hikingEquipment array.</p> <p>Iteratively calls the read methods for all of the Photo objects within the hikingEquipment array in order to retrieve the files and store them in their respective objects for displaying.</p>
Parameters	trailId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString) hikingEquipment.forEach(getPhoto.readPhoto())

3.1.8 LocationDM

Description: LocationDM provides access to the Location table.

Class Diagram

LocationDM
<p>-dm: AccessFirebase -location: Location -locationId: int -sqlString: string -userId: int</p>
<p>+create(userId: int, location: Location): int +delete(userId: int): int +read(userId: int): int +update(userId: int, location: Location): int -verify(location: Location): int</p>

3.1.8.1 Design Elements

3.1.8.1.1 dm: AccessFirebase

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.8.1.2 location: Location

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to upload the Location Table.
Description	Accesses and holds the location of the User, updating in real time if the user has enabled that setting.

3.1.8.1.3 locationID: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Unique id for the Location in the Location Table.
Description	Used to identify the Location stored in the Location Table.

3.1.8.1.4 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.8.1.5 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the userId for the user
Description	Unique integer user ID can be used to retrieve data from User table

3.1.8.1.6 create(userId: int, location: Location): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Create a new instance of a location in the database
Description	<p>Creates a new entry in the location table</p> <p>The data read:</p> <ul style="list-style-type: none">• location_id: int• user_id: int• location: Point <p>Calls verify() before the execution of SQL to ensure that the data to be entered into the table is valid.</p>
Parameters	userId: int, location: Location
Returns	The returned code from the Firebase execution
Functions Called	verify(location: Location) dm.create(sqlString: string): int

3.1.8.1.7 delete(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Deletes the row associated to the passed userId
Description	The row containing the userId is deleted from the location table. A check is done beforehand to ensure that the userId exists in the table
Parameters	userId: int
Returns	The returned code from the Firebase execution
Functions Called	dm.delete(sqlString: string): int

3.1.8.1.8 read(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads the entry containing the provided userId from the location table
Description	<p>Reads the database for the given userId and returns the associated data</p> <p>From the data read: locationid: int location: Point</p>
Parameters	userId: int
Returns	The returned code from the Firebase execution
Functions Called	dm.executeRead(sqlString)

3.1.8.1.9 update(userId: int, location: Location): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Update the location instance.
Description	<p>If verify is successful on the provided location, this method replaces the specified location in the Location table given the passed userId with this data from the provided location:</p> <ul style="list-style-type: none">• location.Point• location.userId <p>This method generates the SQL statement for this command.</p> <p>Updates the location entity in the respective row.</p>
Parameters	userId: int, location: Location
Returns	The returned code from the Firebase execution
Functions Called	Verify(location: Location) dm.executeRead(sqlString)

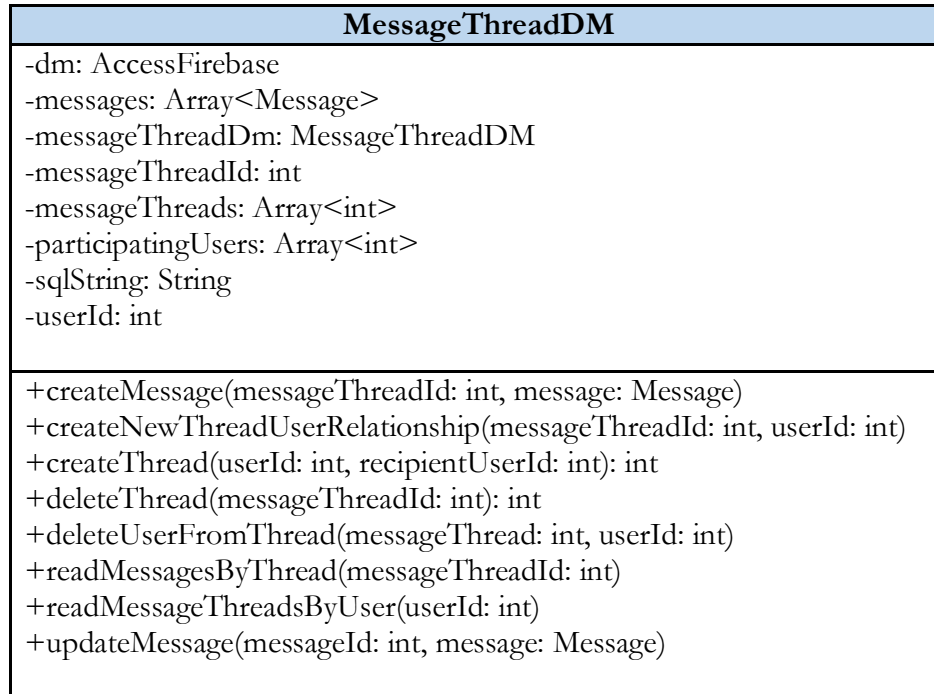
3.1.8.1.10 verify(location: Location): bool

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Verifies that the location provided is valid
Description	verifies that: location.location_id exists location.userId exists location.location exists
Parameters	location: Location
Returns	True if the checks pass, false otherwise.
Functions Called	None

3.1.9 MessageThreadDM

Provides access to a collection of messages sent between users. Manages the Message, Message_Thread, and Message_Thread_Friend tables

Class Diagram



3.1.9.1 Design Elements

3.1.9.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.9.1.2 messages: Array<Message>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store array of message
Description	Stores messages

3.1.9.1.3 messageThreadDm: MessageThreadDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store message thread data management object
Description	Stores message thread data management object

3.1.9.1.4 messageThreadID: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store message thread id
Description	Stores message thread id

3.1.9.1.5 messageThreads: Array<int>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store message thread ID's for message threads that a user is authorized to see
Description	Used to store message thread ID's for message threads that a user is authorized to see

3.1.9.1.6 participatingUsers: Array<int>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store array of integers for users
Description	Stores array of integers for users

3.1.9.1.1 sqlString: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.9.1.2 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store user id
Description	Stores user id

3.1.9.1.3 createMessage(messageThreadId: int, message: Message): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Adds a new message to this message thread
Description	Creates a new row in the Message table with the provided messageThreadId as the parent message thread. Copies over data from the message object that was provided. Generates the SQL statement to be able to accomplish this method.
Parameters	messageThreadId: int, message: Message
Returns	The SQL return code
Functions Called	dm.execute(sqlString): int

3.1.9.1.4 createNewThreadUserRelationship(messageThreadId: int, userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Adds a user to the specified message thread as a participant
Description	Adds a new row to the Message_Thread_Friends table, adding in the messageThreadId and the userId as the data. Generates the SQL statement to be able to accomplish this method.
Parameters	messageThreadId: int, userId: int
Returns	The SQL return code
Functions Called	dm.execute(sqlString): int

3.1.9.1.5 createThread(userId: int, recipientUserId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Creates a new message thread between the provided users
Description	This function adds a new row to the Message_Thread table with the userId as the user_id data, saving the new primary key id as the messageThreadId integer value. Then adds two rows to the Message_Thread_Friends table, with the messageThreadId and the userId and the messageThreadId and the recipientUserId data values. Generates the SQL statement to be able to accomplish this method.
Parameters	userId: int, recipientUserId: int
Returns	The SQL return code
Functions Called	dm.execute(sqlString): int

3.1.9.1.6 deleteThread(messageThreadId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Removes all users from the message thread, preventing them from accessing or seeing the thread any more. This will allow us to keep the thread for our records even if it is “deleted”
Description	Deletes all rows from the Message_Thread_Friend table that match the messageThreadId. Generates the SQL statement to be able to accomplish this method.
Parameters	messageThreadId: int
Returns	The SQL return code
Functions Called	dm.execute(sqlString): int

3.1.9.1.7 deleteUserFromThread(messageThreadId: int, userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Removes the user from the provided thread
Description	Deletes the row from the Message_Thread_Friend table that matches the messageThreadId and the userId. Generates the SQL statement to be able to accomplish this method.
Parameters	messageThreadId: int, userId: int
Returns	The SQL return code
Functions Called	dm.execute(sqlString): int

3.1.9.1.8 readMessagesByThread(messageThreadId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads in all messages from the specified thread.
Description	Reads in all messages from the Message table using the messageThreadId as the select criteria. Saves the Message objects to the messages array. Generates the SQL statement to be able to accomplish this method.
Parameters	messageThreadId: int
Returns	The SQL return code
Functions Called	dm.execute(sqlString): int

3.1.9.1.1 readMessageThreadsByUser(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads in all threads that this user is allowed to view and participate in
Description	Reads all messageThreadId values from the Message_Thread and Message_Thread_Fried table that match the userId and saves the integers into the messageThreads array. Generates the SQL statement to be able to accomplish this method.
Parameters	userId: int
Returns	The SQL return code
Functions Called	dm.execute(sqlString): int

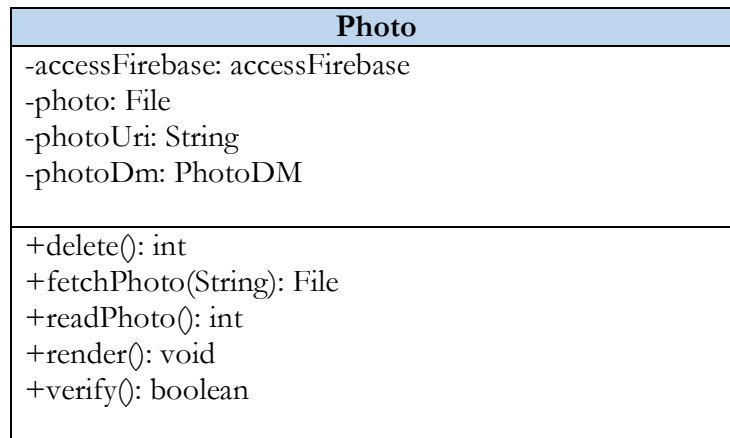
3.1.9.1.1 updateMessage(messageId: int, message: Message): int

ITEM	DESCRIPTION
Type	Function
Purpose	Updates the specified message
Description	Updates the Message table with the data from the given message object. Changes the values of the row in the message table with the messageId primary key to match the object. Generates the SQL statement to be able to accomplish this method.
Parameters	messageId: int, message: Message
Returns	The SQL return code
Functions Called	dm.execute(sqlString): int

3.1.10 Photo

Description: The controller which handles getting the trail's additional information and routing to views.

Class Diagram



3.1.10.1 Design Elements

3.1.10.1.1 accessFirebase: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Access and store data in the firebase database.
Description	This will handle writing and deleting comments to the database.

3.1.10.1.2 photo: File

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	String that contains URI of where the image resides
Description	A String containing an address of an image

3.1.10.1.3 photoUri: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The photoUri holds a String that identifies a particular location within cloud storage.
Description	A String that identifies a photo's location in cloud storage.

3.1.10.1.4 photoDm: PhotoDM

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The photoDM holds the class that allows the database to be queried to obtain the URI for this photo.
Description	The Data Management object that can be used to query the database for the URI.

3.1.10.1.5 delete(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Delete a(n) <i>tableItem</i> from the <i>tableName</i> database
Description	Deletes a single a(n) <i>tableItem</i> from <i>tableName</i> database based upon the ??
Parameters	<i>Parameters based on how you want to delete (see above comment)</i>
Returns	Returns the return code from the SQL execution
Functions Called	verify() dm.execute() dm.commit()

3.1.10.1.6 fetchPhoto(String): File

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Uses HTTP Requests to fetch a Photo File from Cloud storage.
Description	Uses the passed String as the URI to send the request through Firebase to fetch the File object from cloud storage.
Parameters	<i>Takes a String that holds the URI to send the request to in cloud storage.</i>
Returns	Returns the return code from the SQL execution
Functions Called	firebaseAccess.queryFirebase(fetchURI(String))

3.1.10.1.7 readPhoto(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Will retrieve URI attribute
Description	Will set the URI attribute of photo object
Parameters	None
Returns	The return SQL code
Functions Called	photoDM.read()

3.1.10.1.8 render(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Will take the binaryObject of the photo and render it.
Description	Renders the Photo in the view
Parameters	None
Returns	None
Functions Called	None

3.1.10.1.9 verify(): boolean

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Verifies that the attributes are valid
Description	Verifies that the attributes are valid based on the table definition: <ul style="list-style-type: none">• FirebaseAccess != null or blank• PhotoID != 0
Parameters	None
Returns	boolean
Functions Called	None

3.1.10.1.10 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.4.2	Each trail description can display a photo of the trailhead.
3.2.4.2.1	Trail cards shown on the browse screen display a photo of the trailhead and trail name.
3.4.1	Trail Photos load in less than 3 seconds of entering a Browse screen
3.4.2	Trail Photos load in less than 3 seconds of entering a Trail Description screen

3.1.11 PhotoDM

Description: Provides access to the Photo Table.

Class Diagram

PhotoDM
-photoID: int -sql_String: String
+create(): int +read(getPhotoID): int +update(): int +delete(): int -verify(): bool +dm.execute(sql_String)

3.1.11.1 Design Elements

3.1.11.1.1 photoID: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Unique id for a photo
Description	Used to uniquely identify a photo

3.1.11.1.2 photoType: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Unique id type for a photo
Description	Used to uniquely identify for which purpose the photo is used

3.1.11.1.3 photoURI: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	String that contains URI path of where the image resides
Description	A String containing an address of an image

3.1.11.1.4 create(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Create a new instance of a photo in the database
Description	Creates a new entry in the Photo Table. It will use object's attributes to generate a unique ID in the table automatically.
Parameters	None. All the data used is contained in the class's attributes.
Returns	The returned code from the Firebase execution
Functions Called	verify()

3.1.11.1.5 read(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This will read a photo ID
Description	<p>Reads the database for a photo using the photoID. Calls verify(getPhotoID()). This function will use TrailID to perform SQL statement for the photoID from the Trail table. If the return code is correct then SQL statement will execute.</p> <ul style="list-style-type: none">From the data read: photoID photoURI
Parameters	getPhotoID(): int, getTrailID(): int
Returns	The returned code from the Firebase execution
Functions Called	dm.execute(sql_String)

3.1.11.1.6 update(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Update the photo entry
Description	Updates the photo entry in the Photo Table. Calls the verify function to ensure the data is correct before the photo entry is updated.
Parameters	None.
Returns	The returned code from the Firebase execution
Functions Called	verify()

3.1.11.1.7 delete(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function deletes a stored photo
Description	Deletes a photo from the photo table using the given photoID. The deletion will be in one unit of work.
Parameters	None. All the data used is contained in the class's attributes.
Returns	The returned code from the Firebase execution
Functions Called	verify()

3.1.11.1.8 verify(): bool

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Verifies if the new entry is valid.
Description	Verifies that the attributes are valid. photoID != 0
Parameters	None
Returns	Boolean
Functions Called	None

3.1.12 PlantDM

Provides access to the Plant and Plant_Trail tables

PlantDM
-dm: AccessFirebase -plants: Array<Plant> -trailId: int -sqlString: String
+read(trailId: int): int

3.1.12.1 DesignElements

3.1.12.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Enables access to Firebase.
Description	AccessFirebase object for direct database access.

3.1.12.1.2 plants: Array<Plant>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains an array of objects for reference for classes utilizing this one.
Description	Contains an array of Plant objects for storage after deserialization and retrieval from the database.

3.1.12.1.3 trailId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the trailId for the trail.
Description	An integer trail ID that will be the reference point for the junction of these two tables.

3.1.12.1.4 sqlString: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.12.1.5 read(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read Plant objects from the database using the junction table and a trailId.
Description	<p>Accesses the Plant table to retrieve the following data utilizing a join with the Plant_Trail table and the trailId as a condition:</p> <ul style="list-style-type: none"> • plant_id: int • photo: Photo • description: string • is_edible_yn: string • is_dangerous_yn: string <p>Generates the SQL statement to facilitate this join.</p> <p>Deserializes the results into an appropriate number of Plant objects and stores them in plants</p> <p>Iteratively calls the read methods for all of the Photo objects within the plants array in order to retrieve the files and store them in their respective objects for displaying.</p>
Parameters	trailId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString) plants.foreach(getPhoto.readPhoto())

3.1.13 PointOfInterestDM

Provides access to the points of interest in the database

Class Diagram

PointsOfInterestDM
-dm: AccessFirebase -pointsOfInterest: Array<PointOfInterest> -trailId: int -sqlString: String
+create(): int +read(trailId: int): int +update() : int +delete() : int -verify() : bool

3.1.13.1 Design Elements

3.1.13.1.1 accessFirebase: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Access and store data in the firebase database.
Description	This will handle writing and deleting comments to the database.

3.1.13.1.2 pointsOfInterest: Array<PointOfInterest>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Array of interest points
Description	Array containing the points of interest that have been fetched.

3.1.13.1.3 trailId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The ID of the trail that will be used to fetch these points of interest
Description	An integer ID corresponding to a trail.

3.1.13.1.4 sqlString: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	The SQL command String that dm.execute commands use
Description	The SQL command String that dm.execute commands use to create, read, update, or delete data from the database

3.1.13.1.5 create(): int

ITEM	DESCRIPTION
Type	Function
Purpose	Creates a new instance of a point of interest in the database
Description	Creates a new entry in the Points of Interest Table. It will use the objects attributes to generate a unique ID
Parameters	None
Returns	Code from the firebase execution
Functions Called	verify()

3.1.13.1.6 read(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Retrieves all of the PointsOfInterest for a given trailId.
Description	Reads the database for the given trailId using the junction table PointsOfInterest_Trail to gather an array of PointOfInterest objects.
Parameters	trailId: int
Returns	Code from the firebase execution
Functions Called	dm.execute()

3.1.13.1.7 update(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Updates the points of interest entry
Description	Updates the points of interest in the Points of Interest Table. Then calls the verify() function to ensure it is correct
Parameters	None. All information is contained in the class's attributes
Returns	Code from the Firebase execution
Functions Called	verify()

3.1.13.1.8 delete(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Deletes a specific entry in the Points of Interest Table.
Description	Will delete a specific entry in the Points of Interest Table using the pointsOfInterestID in the database
Parameters	None. All information is contained in the class's attributes.
Returns	Code from the firebase execution
Functions Called	verify()

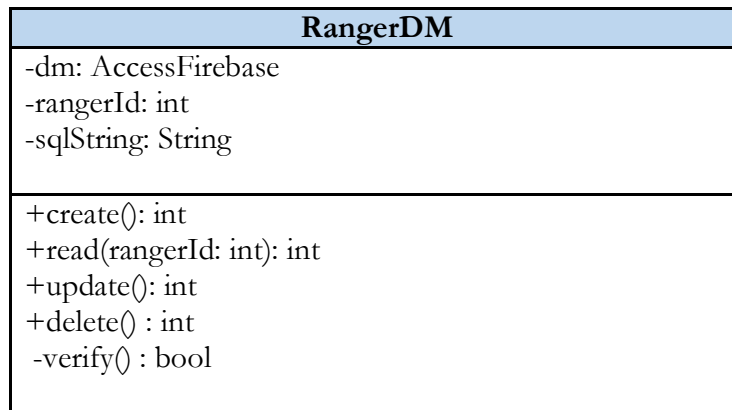
3.1.13.1.9 verify(): bool

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	To verify information in Points of Interest Table
Description	Verifies the attributes are valid. PointOfInterestID !=0 PointOfInterest[] !=0
Parameters	None. All information is contained in the class's attributes.
Returns	Code from firebase execution
Functions Called	None

3.1.14 RangerDM

Description: Provides access to the Ranger table

Class Diagram



3.1.14.1 Design Elements

3.1.14.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	The Data Management object that allows direct access to the database.
Description	This object provides direct access to the database.

3.1.14.1.2 rangerId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Unique id for a ranger
Description	Used to uniquely identify a row from the ranger database

3.1.14.1.3 sqlString: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	A String that will contain the SQL statements to be executed
Description	A String that will contain the SQL statements to be executed

3.1.14.1.4 create(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	To create a new entry in the Ranger table
Description	This will create a new entry in Ranger table. No action is required as the table creates new IDs automatically
Parameters	None. All the data used is contained in the class's attributes.
Returns	The returned code from the Firebase execution
Functions Called	verify(), dm.execute(sqlString)

3.1.14.1.5 read(rangerId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This will read the data from the row corresponding to the rangerId from the table
Description	Reads the database for a ranger using the rangerId <ul style="list-style-type: none">From the data read: first_name last_name phone_number
Parameters	rangerId: int
Returns	The returned code from the Firebase execution
Functions Called	dm.executeRead(sqlString)

3.1.14.1.6 update(id: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Update Ranger table
Description	Updates the Ranger entry in the table. Calls the verify function to ensure the data is correct before the scavenger hunt entry is updated
Parameters	Takes the integer that corresponds to the rangerId to be updated.
Returns	The returned code from the Firebase execution
Functions Called	verify(), dm.execute(sqlString)

3.1.14.1.7 delete(id: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	This function deletes a stored scavenger hunt from the table Ranger
Description	Deletes a stored ranger from the Ranger table using the given rangerId. The deletion will be in one unit of work
Parameters	None. All the data used is contained in the class's attributes.
Returns	The returned code from the Firebase execution
Functions Called	verify(), dm.execute(sqlString)

3.1.14.1.8 verify(): bool

ITEM	DESCRIPTION
Type	Function
Purpose	Verifies if the new scavenger hunt entry is valid
Description	Verifies that the attributes are valid and rangerId != 0
Parameters	None
Returns	True or False
Functions Called	None

3.1.15 ReviewDM

Description: Provides access to the Review table

Class Diagram

ReviewDM
-dm: AccessFirebase -reviews: Array<Review> -sqlString: String -trailId: int -averageRating: float
-readAverageRating(trailId: int): int -read(trailId: int, reviewQuantity: int): int -create(trailId: int, review: Review): int -update(reviewId: int, review: Review): int -delete(reviewId: int): int -verify(review: Review): int

3.1.15.1 Design Elements

3.1.15.1.1 dm: AccessFirebase

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store data
Description	This object provides direct access to the database.

3.1.15.1.2 reviews: Array<Review>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store array of reviews
Description	Stores array of reviews to be which are fetched from database

3.1.15.1.3 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A String that will contain the SQL statements to be executed
Description	A String that will contain the SQL statements to be executed

3.1.15.1.4 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store trail id integer
Description	Trail id integer is a unique trail identifier

3.1.15.1.5 averageRating: float

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store average float of ratings for specific trail
Description	A float that is represents average ratings for the trail

3.1.15.1.6 readAverageRating(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This will read from the Review table to aggregate the reviews from all reviews associated with the provided trailId
Description	<p>Reads the Review table, selecting only the rating column from all reviews that match the trailId.</p> <p>From the data read:</p> <ul style="list-style-type: none"> • rating <p>Generates an SQL statement to retrieve the data.</p> <p>Takes all of the returned values, adding them together, then dividing them by the total quantity of ratings retrieved. This is then saved in the variable averageRating.</p>
Parameters	trailId: int
Returns	The returned code from the Firebase execution
Functions Called	dm.executeRead(sqlString)

3.1.15.1.7 read(trailId: int, reviewQuantity: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This will read from the Review table, retrieving the most recent reviews for the provided trailId at most of the quantity provided
Description	<p>Reads the Review table, retrieving all reviews for the trailId.</p> <p>From the data read:</p> <ul style="list-style-type: none"> • review_id: int • comment: string • rating: int • user_id: int • trail_id: int • created_at: datetime • modified_at: datetime <p>Will save only a quantity of reviews to the reviews object that equal reviewQuantity, this is determined by that number of reviews that were most recently created_at or modified_at (priority to modified_at).</p> <p>Generates an SQL statement to retrieve the data.</p>
Parameters	trailId: int; reviewQuantity: int
Returns	The returned code from the Firebase execution
Functions Called	dm.executeRead(sqlString)

3.1.15.1.8 create(trailId: int, review: Review): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to create a new review for the provided trail.
Description	<p>Creates a new review for this trailId using the provided review object.</p> <p>Creates the SQL statement for this command.</p> <p>Calls verify before calling the SQL statement, if verify returns true, proceeds with the execution, else this method fails.</p>
Parameters	trailId: int, review: Review
Returns	The return code from FireBase execution
Functions Called	verify(review: Review) dm.execute(sqlString)

3.1.15.1.9 update(reviewId: int, review: Review): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Make a change in a particular Review row at the provided reviewId.
Description	<p>If verify is successful on the provided review, this method replaces the specified review in the Review table given its reviewId with this data from the provided review:</p> <ul style="list-style-type: none">• rating• comment <p>This method generates the SQL statement for this command.</p>
Parameters	reviewId: int, review: Review
Returns	The return code from FireBase execution
Functions Called	verify(review: Review) dm.execute(sqlString)

3.1.15.1.10 delete(reviewId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Removes the association between the review and the trail, effectively deleting it, but retaining the review itself for our records.
Description	Removes the trail_id from the specified comment at reviewId in the Review table. Generates the SQL command to be able to perform this command.
Parameters	reviewId: int
Returns	The return code from FireBase execution
Functions Called	dm.execute(sqlString)

3.1.15.1.11 verify(review: Review): int

ITEM	DESCRIPTION
Type	Function
Purpose	Verifies that the review provided is appropriate and valid
Description	verifies that: review.comment does not contain swear words review.trailId exists review.userId exists review.rating is within 1 and 5
Parameters	review: Review
Returns	True if the provided object passes the checks, false otherwise.
Functions Called	None

3.1.16 ScavengerHuntDM

Description: Provides access to the ScavengerHunt table

Class Diagram

ScavengerHuntDM
-dm: AccessFirebase -scavengerHuntId: int -sqlString: String -pointsOfInterest: Array<PointOfInterest> -description: string -photo: Photo
+read(scavengerHuntId: int): int

3.1.16.1 Design Elements

3.1.16.1.1 dm: AccessFirebase

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.16.1.2 scavengerHuntId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Unique id for a scavenger hunt
Description	Used to uniquely identify scavenger hunt

3.1.16.1.3 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A String that will contain the SQL statements to be executed
Description	A String that will contain the SQL statements to be executed

3.1.16.1.4 pointsOfInterest: Array<PointOfInterest>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Holds an array of point of interest Objects
Description	An array of objects that will contain points of interest

3.1.16.1.5 description: string

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A description that describes the scavenger hunt itself to users
Description	A String that tells the user what to look for during the scavenger hunt.

3.1.16.1.6 photo: Photo

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	To hold a photo object
Description	An objects which holds a photo string and File

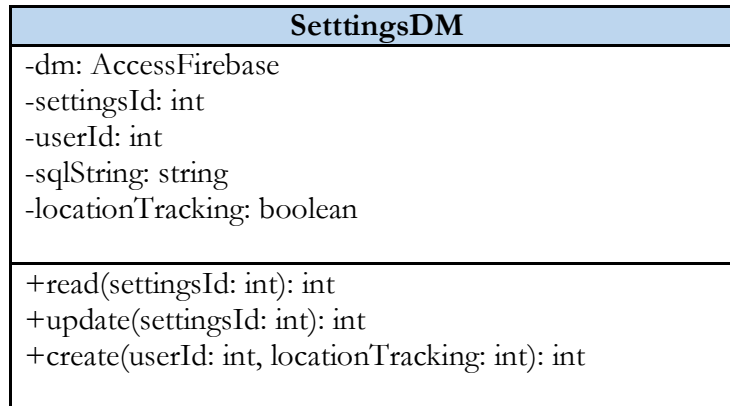
3.1.16.1.7 read(scavengerHuntId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This will read from the ScavengerHunt table
Description	<p>Reads the database for a scavenger hunt using the scavengerHuntId. Generates SQL statement to retrieve the data.</p> <p>From the data read:</p> <ul style="list-style-type: none">• scavengerHuntId• description• photo <p>Accesses the Points of Interest table to retrieve the following data utilizing a join with the Scavenger_Hunt_POI table and the scavengerHuntId as a condition:</p> <p>From the data read:</p> <ul style="list-style-type: none">• pointOfInterestId: int• photoId: int• location: Point• description: string <p>Deserializes the results into an appropriate number of PointOfInterest objects and stores them in pointOfInterest</p> <p>Iteratively calls the read methods for all of the Photo objects within the pointOfInterest array in order to retrieve the files and store them in their respective objects for displaying.</p>
Parameters	scavengerHuntId: int
Returns	The returned code from the Firebase execution
Functions Called	dm.executeRead(sqlString)

3.1.18 SettingsDM

Description: Enables access to the Settings table

Class Diagram



3.1.18.1 Design Elements

3.1.18.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.18.1.2 settingsId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Identify a given User's settings.
Description	Unique ID for a User's settings.

3.1.18.1.3 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Identify a given User.
Description	Unique ID for a User

3.1.18.1.4 sqlString: string

ITEM	DESCRIPTION
Type	Attribute
Purpose	A String that will contain the SQL statements to be executed
Description	A String that will contain the SQL statements to be executed

3.1.18.1.5 locationTracking: bool

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Determine if the user wants to permit location tracking.
Description	Boolean indicating whether the user wants location tracking on.

3.1.18.1.6 readSettings(settingsId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This will read from the Settings table.
Description	Reads the user's settings for location. From data read: <ul style="list-style-type: none">• locationTracking: bool Using the settingsId within a SQL statement string.
Parameters	settingsId: int
Returns	The return code from FireBase execution.
Functions Called	dm.executeRead(sqlString)

3.1.18.1.7 updateSettings(settingsId: int, locationTracking: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This will update a row in the Settings table.
Description	If verify is successful for the userId, the new state of locationTracking is passed to an SQL statement for update in the database.
Parameters	settingsId: int
Returns	The return code from FireBase execution
Functions Called	verify(locationTracking: bool) dm.execute(sqlString)

3.1.18.1.8 createSettings(userId: int, locationTracking: bool): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This will create a row in the Settings table.
Description	If verify is successful for the userId, the initial state of locationTracking and userId is passed to an SQL statement for new row creation in the database. The settingsId is automatically generated for the row.
Parameters	userId: int locationTracking: int
Returns	The return code from FireBase execution.
Functions Called	dm.execute(sqlString)

3.1.19 TrailDM

Description: Provides access to the Trail table

TrailDM
-dm: AccessFirebase -photos: Array<Photo> -pointOfInterestDm: PointOfInterestDM -routePoints: Array<Point> -sqlString: String -trail: Trail -trailId: int -trails: Array<Trail>
+read(trailId: int): int +readPhotos(trailId: int): int +readRoutePoints(trailId: int): int +readTrailsByFilter(location: Point, overlookIsRequired: boolean, waterfallIsRequired: boolean): int

3.1.19.1 Design Elements

3.1.19.1.1 dm: AccessFirebase

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.19.1.2 photos: Array<Photo>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	An Array of Photo objects for displaying and asynchronous retrieval
Description	An Array of Photo objects for displaying and asynchronous retrieval

3.1.19.1.3 pointOfInterestDm: PointOfInterestDM

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A PointOfInterestDM for the readTrailsByFilter method
Description	A PointOfInterestDM for the readTrailsByFilter method

3.1.19.1.1 routePoints: Array<Point>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	An array of Points for demonstrating the flow of the trail
Description	An array of Points for demonstrating the flow of the trail

3.1.19.1.2 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.19.1.3 trail: Trail

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The trail object that stores an individually-retrieved trail
Description	The trail object that stores an individually-retrieved trail from the Trail database

3.1.19.1.4 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The ID of the trail to be retrieved
Description	The integer ID of the trail to be retrieved

3.1.19.1.1 trails: Array<Trail>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	An array of trails that work with the getTrailsByFilter method
Description	An array of trails that work with the getTrailsByFilter method

3.1.19.1.2 read(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read trail information
Description	Accesses a Trail table to retrieve data by trailId. Generates an SQL statement to fetch the row that corresponds to the trailId. Fetches the following data: <ul style="list-style-type: none">• trail_head: Point• rangerId: int• map_id: int• additional_info_id: int• scavenger_hunt_id: int• trail_name: String• description: String• disability_accessible_yn: bool• num_steps: int• length_miles: float
Parameters	trailId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

3.1.19.1.3 readPhotos(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to retrieve all of the Photos from the database based on the provided trailId
Description	<p>Accesses the Photo_Trail table to make a join from it to the Photo table, using trailId as the condition. This retrieves all photo object rows associated with the trail, including the following data:</p> <ul style="list-style-type: none">• URI: string• photoId: int <p>Deserializes the data into the photos array of Photo objects. Calls each of their respective read objects to pull the files into memory.</p> <p>Generates the SQL statement to be executed.</p>
Parameters	trailId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString) photo.foreach(read())

3.1.19.1.4 readRoutePoints(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to retrieve all of the routePoints for the trail
Description	<p>Accesses the Trail_Route_Points table to make a join between it and the Route_Points table conditional upon the Trail_Route_Points rows that match the trailId to retrieve the following data from Route_Points:</p> <ul style="list-style-type: none">• location: Point <p>All of the Point objects are saved into the routePoints array.</p> <p>Generates the SQL statement to be executed.</p>
FParameters	trailId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

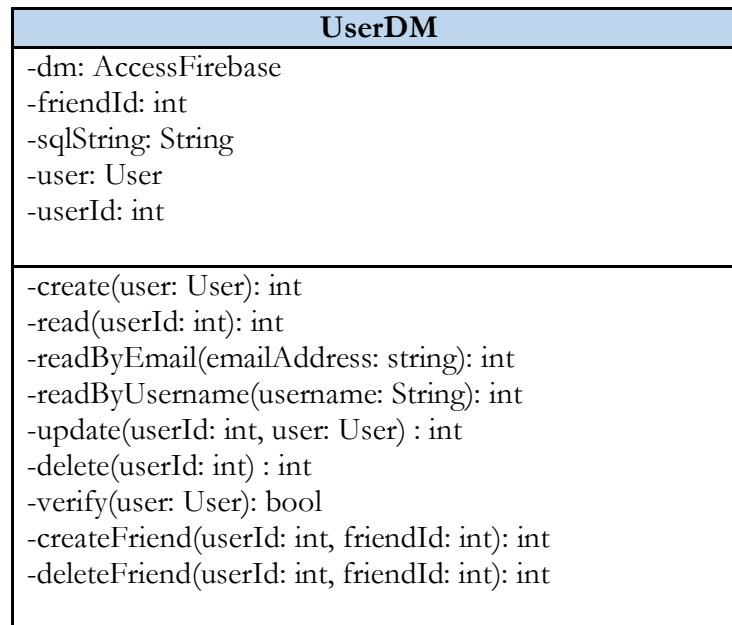
3.1.19.1.5 readTrailsByFilter(location: Point, overlookIsRequired: boolean, waterfallIsRequired: boolean): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to retrieve specific Trails based on some arbitrary constraints
Description	<p>Accesses a Trail table to retrieve data based on trails where the trail_head is located within 50 miles of the provided location (calculated using Euclidean distance). Generates an SQL statement to fetch the row that corresponds to the trailId. Fetches the following data:</p> <ul style="list-style-type: none"> • trail_head: Point • rangerId: int • map_id: int • additional_info_id: int • scavenger_hunt_id: int • trail_name: String • description: String • disability_accessible_yn: bool • num_steps: int • length_miles: float <p>Ends execution here if overlookIsRequired is false and waterfallIsRequired is false.</p> <p>Otherwise, once this list of trails has been deserialized, this method will use PointOfInterestDM to look up all of the PointOfInterest for each of the retrieved trails, retrieving only:</p> <ul style="list-style-type: none"> • POI_Type: int <p>If waterfallIsRequired is true, and for each trail, there exists no PointOfInterest that has a POI_Type that is equal to the types representing waterfalls, that trail is removed from the trails array.</p> <p>If overlookIsRequired is true, and for each trail, there exists no PointOfInterest that has a POI_Type that is equal to the types representing overlook, that trail is removed from the trails array.</p> <p>Generates the SQL statement to be executed.</p>
Parameters	location: Point, overlookIsRequired: boolean, waterfallIsRequired: boolean
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString)

3.1.20 UserDM

Description: UserDM provides access to the User table

Class Diagram



3.1.20.1 Design Elements

3.1.20.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this class to access the database directly.
Description	Allows this class to access the database directly.

3.1.20.1.2 friends: Array<User>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store array of user objects
Description	Stores array of user objects

3.1.20.1.3 friendId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store an int to represent friends to be added or removed
Description	Stores an integer representing the friend to be added or removed

3.1.20.1.4 sqlString: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.20.1.5 user: User

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the user object for retrieval
Description	The User object that will be used for the storage of retrieved data

3.1.20.1.6 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the userId for the user object
Description	Unique integer user ID can be used to retrieve data from User table

3.1.20.1.7 create(user: User): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Creates a new User from the provided object
Description	<p>Adds a new row to the User table using the data provided in the user object.</p> <p>Creates the SQL statement for this command.</p> <p>Calls verify before calling the SQL statement, if verify returns true, proceeds with the execution, else this method fails</p>
Parameters	user: User
Returns	The return code from FireBase execution
Functions Called	verify(user: User) dm.execute(sqlString)

3.1.20.1.8 read(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read in data from the User table using the provided userId
Description	<p>Retrieves the user using the provided userId integer. Retrieves:</p> <ul style="list-style-type: none">• user_id: int• photo_id: int• settings_id: int• first_name: string• last_name: string• about_me: string• username: string• password: string• email_address: string• age: int• weight: float• height: float <p>Additionally, retrieves an array of friends using this userId from the Friends table, deserializing the select statement as it is retrieved into the user.friends array, selecting the:</p> <ul style="list-style-type: none">• friend_id: int <p>Also retrieves the favorite trails of the user from the User_Trail table, retrieving an array of integers to be stored into the user.favoriteTrails array by selecting all rows that feature this user's userId.</p> <ul style="list-style-type: none">• trail_id: int <p>Creates the SQL statement for this command</p>
Parameters	userId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString) dm.execute(sqlString)

3.1.20.1.9 readByEmail(emailAddress: String): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read in data from the User table using the provided email address
Description	<p>Retrieves the user using the provided emailAddress String.</p> <p>Retrieves:</p> <ul style="list-style-type: none">• user_id: int• photo_id: int• settings_id: int• first_name: string• last_name: string• about_me: string• username: string• password: string• email_address: string• age: int• weight: float• height: float <p>Additionally, retrieves an array of friends using this userId from the Friends table, deserializing the select statement as it is retrieved into the user.friends array, selecting the:</p> <ul style="list-style-type: none">• friend_id: int <p>Also retrieves the favorite trails of the user from the User_Trail table, retrieving an array of integers to be stored into the user.favoriteTrails array by selecting all rows that feature this user's userId.</p> <ul style="list-style-type: none">• trail_id: int <p>Creates the SQL statement for this command.</p>
Parameters	emailAddress: String
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString) dm.execute(sqlString)

3.1.20.1.10 readByUsername(username: String): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read in data from the User table using the provided username
Description	<p>Retrieves the user using the provided username String.</p> <p>Retrieves:</p> <ul style="list-style-type: none">• user_id: int• photo_id: int• settings_id: int• first_name: string• last_name: string• about_me: string• username: string• password: string• email_address: string• age: int• weight: float• height: float <p>Additionally, retrieves an array of friends using this userId from the Friends table, deserializing the select statement as it is retrieved into the user.friends array, selecting the:</p> <ul style="list-style-type: none">• friend_id: int <p>Also retrieves the favorite trails of the user from the User_Trail table, retrieving an array of integers to be stored into the user.favoriteTrails array by selecting all rows that feature this user's userId.</p> <ul style="list-style-type: none">• trail_id: int <p>Creates the SQL statement for this command.</p>
Parameters	username: String
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString) dm.execute(sqlString)

3.1.20.1.11 update(userId: int, user: User) : int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Updates a user.
Description	<p>If verify is successful on the provided user, this method replaces the specified row in the User table given its userId with this data from the provided user object:</p> <ul style="list-style-type: none">• user_id: int• photo_id: int• settings_id: int• first_name: string• last_name: string• about_me: string• username: string• password: string• email_address: string• age: int• weight: float• height: float <p>This method generates the SQL statement for this command.</p>
Parameters	userId: int, user: User
Returns	The return code from FireBase execution
Functions Called	verify(user: User) dm.execute(sqlString)

3.1.20.1.12 delete(userId: int) : int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Removes the row from the User table corresponding to the provided userId
Description	<p>Removes the row from the User table corresponding to the provided userId. Generates the SQL command to be able to perform this command.</p>
Parameters	userId: int
Returns	The return code from FireBase execution
Functions Called	dm.execute(sqlString) verify(userId: int)

3.1.20.1.13 verify(user: User): bool

ITEM	DESCRIPTION
Type	Function
Purpose	Verifies that the user provided is appropriate and valid
Description	verifies that: first_name contains no swear words last_name contains no swear words about_me contains no swear words username contains no swear words email_address is a valid email address returns true if all of the above are valid, and false otherwise.
Parameters	user: User
Returns	True if the comment passes the checks, false otherwise.
Functions Called	None

3.1.20.1.14 createFriend(userId: int, friendId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Used to create a new friend relationship in the Friend table.
Description	Creates a new row in the Friends table provided that the composite key userId and friendId is not to be found there. This will check both userId and friendId and flipping them for each other. Creates the SQL statement for this command.
Parameters	userId: int, friendId: int
Returns	The return code from FireBase execution
Functions Called	dm.execute(sqlString)

3.1.20.1.15 deleteFriend(userId: int, friendId: int) : int

ITEM	DESCRIPTION
Type	Function
Purpose	Removes a friend relationship from the Friend table
Description	Deletes a row in the Friends table provided that the composite key userId and friendId is found there. This will check both userId and friendId and flipping them for each other.

	Generates the SQL command to be able to perform this command.
Parameters	userId: int, friendId: int
Returns	The return code from FireBase execution
Functions Called	dm.execute(sqlString)

3.1.21 WildlifeDM

Provides access to the Wildlife and Wildlife_Trail tables

WildlifeDM
-dm: AccessFirebase -sqlString: String -trailId: int -wildlife: Array<Wildlife>
+read(trailId: int): int

3.1.21.1 DesignElements

3.1.21.1.1 dm: AccessFirebase

ITEM	DESCRIPTION
Type	Attribute
Purpose	Enables access to Firebase.
Description	AccessFirebase object for direct database access.

3.1.21.1.2 sqlString: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	A string for the execution of SQL commands.
Description	A string for the execution of SQL commands.

3.1.21.1.3 trailId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the trailId for the trail.
Description	An integer trail ID that will be the reference point for the junction of these two tables.

3.1.21.1.4 wildlife: Array<Wildlife>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains an array of objects for reference for classes utilizing this one.
Description	Contains an array of Wildlife objects for storage after deserialization and retrieval from the database.

3.1.21.1.5 read(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to read Wildlife objects from the database using the junction table and a trailId.
Description	<p>Accesses the Wildlife table to retrieve the following data utilizing a join with the Wildlife_Trail table and the trailId as a condition:</p> <ul style="list-style-type: none">• wildlife_id: int• photo: Photo• description: string• warning_signs: string• wildlife_name: string <p>Generates the SQL statement to facilitate this join. Deserializes the results into an appropriate number of Wildlife objects and stores them in wildlife</p> <p>Iteratively calls the read methods for all of the Photo objects within the wildlife array in order to retrieve the files and store them in their respective objects for displaying.</p>
Parameters	trailId: int
Returns	The return code from FireBase execution
Functions Called	dm.executeRead(sqlString) wildlife.foreach(getPhoto.readPhoto())

3.2 Home

The Home view contains the core views and controllers that the app will default to when it is first opened. These views are either navigation-based or overarching settings-based.

3.2.1 HomeController

Description: A controller that facilitates navigation to all of the other parts of the app. This is the landing page for when the app is first started. The HomeController also takes care of the NavBar.

Class Diagram

HomeController
-settings: Settings -user: User
+openHomeFirstTime(): void +openView(View): void

3.2.1.1 Design Elements

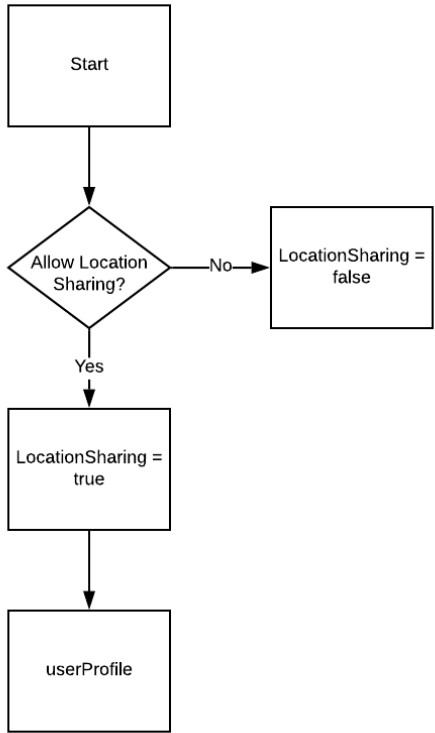
3.2.1.1.1 settings: Settings

ITEM	DESCRIPTION
Type	Attribute
Purpose	Holds the Settings object for the LocationSharing settings modification.
Description	Loads the Settings saved locally to the phone.

3.2.1.1.2 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	Holds the session's user object
Description	This user object is the basis for many create statements. This contains the current user of the app as a session identifier.

3.2.1.1.3 openHomeFirstTime(): void

ITEM	DESCRIPTION
Type	Function
Purpose	When a user navigates to the Home Screen for the first time, this method will guide the user through a help dialogue.
Description	<p>firstTime() pops up a modal that asks for application permissions for Location Sharing. If the user says yes, the Settings for LocationSharing is enabled to true. If the user says no, the Settings for LocationSharing is disabled to false. This setting is also changed in Firebase for the User as well as enabled for the phone's GPS hardware if it is enabled. This method then spotlights the userProfile.</p>  <pre> graph TD Start([Start]) --> Decision{Allow Location Sharing?} Decision -- No --> False[LocationSharing = false] Decision -- Yes --> True[LocationSharing = true] True --> UserProfile[userProfile] </pre>
Parameters	None
Returns	None
Functions Called	None

3.2.1.1.4 openView(View): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Opens a new View, enables the user to navigate through the app.
Description	<p>Opens the provided View. Possible Views that can be opened from the Home Screen include:</p> <ul style="list-style-type: none">• BrowseController• Settings• FilterController• LoginController• EditProfileController• AllTrailsController• FriendsController <p>This method will be static, allowing it to be referenced from other contexts. Those contexts may provide Controllers to change the view to that are not listed here, but are still valid.</p>
Parameters	Takes a View object that the app will then navigate to.
Returns	None
Functions Called	None

3.2.1.1.5 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Provide a Home screen.
3.2.1.2	The Bottom Navigation Panel has a House icon, Browse icon, Chat icon, and Map icon.
3.2.1.5.	Tapping the house icon navigates the User to the Home screen.
3.2.2.3.5.1	If the app navigates the User to the Home screen for the first time, the app displays a Modal asking for permissions to track the User's location.
3.2.2.3.5.2	After 2 seconds of landing on the Home screen for the first time, the app spotlights the User's profile icon at the top right of the Home screen
3.2.2.3.5.3	The spotlight displays a message indicating to the User to finish completing the User's profile and is dismissed with a tap on the screen.
3.2.2.3.5.4	A second spotlight appears on the Browse button icon indicating that a User may browse Trails here and is also dismissed by tapping the screen.

3.2.2 SettingsController

Description: A controller that contains the details the user can change on his profile. User can enable Location Tracking.

Class Diagram

SettingController
-locationSetting: boolean -settingsDm: SettingsDM -settingsId: int -userId: int
+createSettings(userId: int, locationSharingSetting: boolean): int +readSettings(settingsId: int): int +updateSettings(settingsId: int, locationSharingSetting: boolean): int

3.2.2.1 Design Elements

3.2.2.1.1 locationSetting: boolean

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores the state of the location tracking setting.
Description	Boolean storing the state of the location tracking boolean setting.

3.2.2.1.2 settingsDm: SettingsDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allow the class to interact with the Settings table of the database.
Description	Connection to the Settings table in the database.

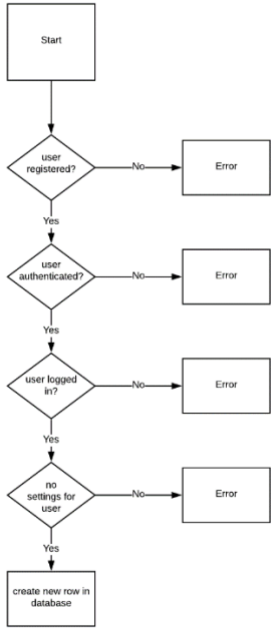
3.2.2.1.3 settingsId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores the integer that represents the setting's ID in the table.
Description	Integer storing the id for a User's settings.

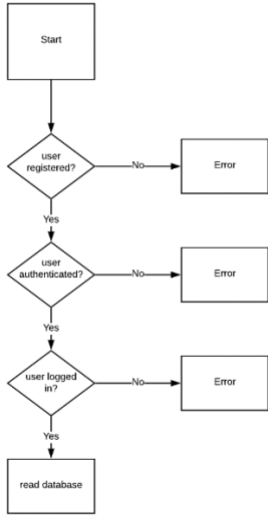
3.2.2.1.4 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores the integer that represents the User's ID.
Description	Store the integer that represents the User's ID.

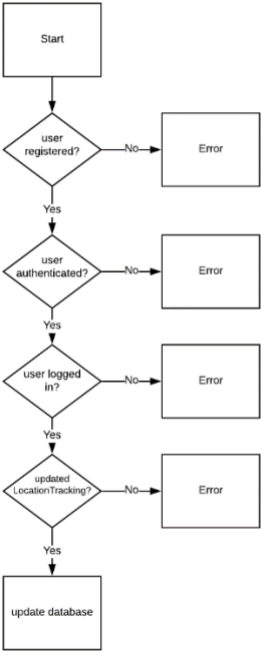
3.2.2.1.5 createSettings(userId: int, locationSetting: boolean): int

ITEM	DESCRIPTION
Type	Function
Purpose	Initialize the User's settings in the database.
Description	<p>Access database to create the user's setting for location tracking</p>  <pre> graph TD Start([Start]) --> Registered{user registered?} Registered -- No --> Error1[Error] Registered -- Yes --> Authenticated{user authenticated?} Authenticated -- No --> Error2[Error] Authenticated -- Yes --> LoggedIn{user logged in?} LoggedIn -- No --> Error3[Error] LoggedIn -- Yes --> Settings{no settings for user} Settings -- No --> Error4[Error] Settings -- Yes --> CreateRow[create new row in database] </pre>
Parameters	userId: int locationSharingSetting: boolean
Returns	The return SQL code
Functions Called	settingsDm.update(userId: int, locationSharingSetting: boolean)

3.2.2.1.6 readSettings(settingsId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Retrieve the User's settings in the database.
Description	<p>Accesses database to read the location tracking setting in the database.</p>  <pre> graph TD Start([Start]) --> Registered{user registered?} Registered -- No --> Error1[Error] Registered -- Yes --> Authenticated{user authenticated?} Authenticated -- No --> Error2[Error] Authenticated -- Yes --> LoggedIn{user logged in?} LoggedIn -- No --> Error3[Error] LoggedIn -- Yes --> ReadDB[read database] </pre>
Parameters	settingsId: int
Returns	The return SQL code
Functions Called	settingsDm.read(settingsId: int)

3.2.2.1.7 updateSettings(settingsId: int, locationSharingSetting: boolean): int

ITEM	DESCRIPTION
Type	Function
Purpose	Updates the User's settings in the database.
Description	<p>Accesses database to change the LocationTracking setting in the database.</p>  <pre> graph TD Start([Start]) --> Registered{user registered?} Registered -- No --> Error1[Error] Registered -- Yes --> Authenticated{user authenticated?} Authenticated -- No --> Error2[Error] Authenticated -- Yes --> LoggedIn{user logged in?} LoggedIn -- No --> Error3[Error] LoggedIn -- Yes --> Updated{updated LocationTracking?} Updated -- No --> Error4[Error] Updated -- Yes --> UpdateDB[update database] </pre>
Parameters	settingsId: int locationSharingSetting: Boolean
Returns	The return SQL code
Functions Called	settingsDm.update(settingsId: int, locationSharingSetting: boolean)

3.2.2.1.8 Design Concerns

SRS	Content
3.2.1.1	Provide a Settings screen.
3.2.3.1	If the User did not deselect app notifications on the User's device settings, the app will send a Push Notification every Thursday of each week that suggests a list of hiking Trails that the User has never started.
3.2.6.8.1	By default, location tracking toggle in the Settings screen shall be OFF

3.3 Login and Registration

This view contains authentication and user data controllers and classes. The user can create their account through the RegisterController, log in through the LoginController, or renew their password through the ForgotMyPasswordController.

3.3.1 ForgotMyPasswordController

Description: Manages the password recovery process

Class Diagram

ForgotMyPasswordController
-user: User -userDm: UserDM -userId: int -twoFactorAuth: TwoFactorAuthentication
+sendRecoveryEmail(user.getEmailAddress()): int +updateUser(userId: int, user: User): int

3.3.1.1 Design Elements

3.3.1.1.1 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	To contain personal information from the user.
Description	A attribute that contains personal information from the user.

3.3.1.1.2 userDm: UserDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used for storing user information in a data management object
Description	Stores user object attributes from the controller to create a user

3.3.1.1.3 emailAddress: string

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store email address
Description	Stores email address in a variable to lookup an email address in the database

3.3.1.1.4 twoFactorAuth: TwoFactorAuthentication

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store two factor authentication entity
Description	Stores two factor authentication attributes in this controller

3.3.1.1.5 sendRecoveryEmail(emailAddress: string): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Sends a recovery email to the user
Description	Sends a recovery email to the user if the email address provided matches any of the users in the table. Recovery email will contain a link to reset password. Holds in the session a two-factor authentication code for the user to provide when they receive an email.
Parameters	emailAddress: string
Returns	The results of the SQL code
Functions Called	validateInputs(): boolean userDm.readByEmail(emailAddress: string): int twoFactorAuth.set(emailAddress: string): void

3.3.1.1.6 updateUser(userId: int, user: User): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Attempts to make the potential user a real user.
Description	<p>Takes the properties stored in the user object and validates them, if they pass validation, updates the User table with the new user using the userDm object. If the User enters in the correct code they received, the they will be prompted to enter a new password. If the User does not enter the correct code, the app re-prompts the User for the code up to four times.</p> <p>When the user's password has been updated in the database it will switch to Login Controller view</p>
Parameters	user: User
Returns	The results of the SQL code
Functions Called	validateInputs(): boolean User.hashPassword(): string twoFactorAuth.startProcess(): void user.setPassword(password: string): string userDm.update(user: User): int HomeController.openView(LoginController)

3.3.1.1.7 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Provide a "Forgot my Password" screen.
3.2.2.2.1	Tapping the "Forgot my password" button will navigate the User to the "Forgot my Password" screen
3.2.2.2.2	The "Forgot my Password" screen will prompt the User to enter their email address.
3.2.2.2.2.1	Provided a registered email address, a Two-Factor Authentication email is sent to the User with a 6-digit numeric code, which the screen will prompt for.
3.2.2.2.2.1.1	If the email address is not registered, a message indicating that the email address has not been registered displays.
3.2.2.2.2.2	If the User enters in the correct code they received, the they will be prompted to enter a new password.
3.2.2.2.2.2.1	If the User does not enter the correct code, the app re-prompts the User for the code up to four times.
3.2.2.2.2.2.2	Failure to enter the correct code four times will send the user back to the login screen.

3.3.2 LoginController

Description: Controller for the login part of the Login screen

Class Diagram

LoginController
-user: User -userId: int
+attemptLogIn(string: password): boolean -goToRegisterController(): void -read(userId: int): int

3.3.2.1 Design Elements

3.3.2.1.1 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	To contain personal information from the user.
Description	A attribute that contains personal information from the user.

3.3.2.1.2 UserId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	To contain a number associated with the user
Description	An integer that is associated with a given user.

3.3.2.1.3 attemptLogIn(string: password): boolean

ITEM	DESCRIPTION
Type	Function
Purpose	To attempt to login to a particular user with a password.
Description	You put a password in, and if the function returns true, you log into your user. Additionally, we need to hash the password and then compare the hashed string with the string in the database. Read function will retrieve user object and compare its password encoded parameter with the password that was provided in the input statement. If these two match then HomeController will be instantiated.
Parameters	password: string
Returns	Boolean
Functions Called	user.hashPassword(password: string): string read(username: string): int HomeController(user: User)

3.3.2.1.4 goToRegisterController(): void

ITEM	DESCRIPTION
Type	Function
Purpose	To navigate to a register controller
Description	This function will take us to the register controller when user click “Create account” button
Parameters	None
Returns	None
Functions Called	HomeController.openView(RegisterController)

3.3.2.1.5 read(username: string): int

ITEM	DESCRIPTION
Type	Function
Purpose	To access information from the database using the UserId.
Description	the function uses the userID to access information from the database about the user.
Parameters	userID: int
Returns	The return value for the SQL code execution
Functions Called	userDm.read(userID: int): int

3.3.2.1.6 Design Concerns

SRS	Content
3.2.1.1	Provide a login screen.
3.2.2.1	Two text fields for Username and password are displayed on the Login screen
3.2.2.1.1	The password field requires eight or more characters that include at least one capital letter, one special character, and one number.
3.2.2.1.2	Inputting the correct Username and password will navigate the User to the Home screen
3.2.2.2.3	If the User types a valid password according to the guidelines from 3.2.2.1.1, the app will display a message indicating that the User's password is correct
3.2.2.2.4	The app displays a "Return to the login page" button. When pressed, the app navigates the User back to the Login screen

3.3.3 RegisterController

Description: Controller for the Register Screen.

Class Diagram

RegisterController
-user: User -userDm: UserDM
+validateInputs(): boolean +createUser(user: User): int

3.3.3.1 Design Elements

3.3.3.1.1 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used for storing user inputs in the user object to create a user in the database
Description	Stores user attributes of a user object to initiate and complete registration

3.3.3.1.2 userDm: UserDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used for storing user information in a data management object
Description	Stores user object attributes from the controller to create a user

3.3.3.1.3 validateInputs(): boolean

ITEM	DESCRIPTION
Type	Function
Purpose	Checks whether or not input data from the view is valid
Description	<ul style="list-style-type: none">- The email address input should be a valid email address (i.e. have an '@' symbol, etc.)- The other String inputs should not have any sort of injection potential (SQL/HTML/JS) or other security problems. <pre>{ emailIsValid = regex.match(this.email, emailRegex) this.firstName = this.firstName.strip() firstNameIsValid = this.firstName.isAlpha() ... (all Attributes will be verified similarly) return emailIsValid && firstNameIsValid && lastNameIsValid && passwordIsValid && ETC }</pre>
Parameters	None – acts on internal class properties
Returns	Returns true if inputs are valid and false if they are not
Functions Called	None

3.3.3.1.4 createUser(user: User): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Attempts to make the potential user a real user.
Description	Takes the properties stored in the user object and validates them, if they pass validation, updates the User table with the new user using the userDm object.
Parameters	user: User
Returns	The results of the SQL code
Functions Called	validateInputs(): boolean User.hashPassword(): string userDm.createUser(user: User): int

3.3.3.1.5 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Provide a Register screen.
3.2.2.3.1	Tapping the “Register for an account” button navigates the User to the Register screen
3.2.2.3.2	The Register screen shall prompt the User for their First name, Last name, Email address, and a valid password (see 3.2.2.1.1).
3.2.2.3.4	Tapping the “Register” button without providing all the information in 3.2.2.3.2 causes the app to re-prompt the user for that information.
3.2.2.3.5	Tapping on the “Register” button with all the fields in 3.2.2.3.2 filled out will navigate the User to the Home screen.

3.3.4 TwoFactorAuthentication

Description: A security mechanism that requires confirmation via a mobile device to process logins. It helps ensure that nobody, but the proper user will be able to log into their account.

Class Diagram

TwoFactorAuthentication
-emailAddress: String -pinCode: String
+pinIsValid(): boolean +startProcess(): void

3.3.4.1 Design Elements

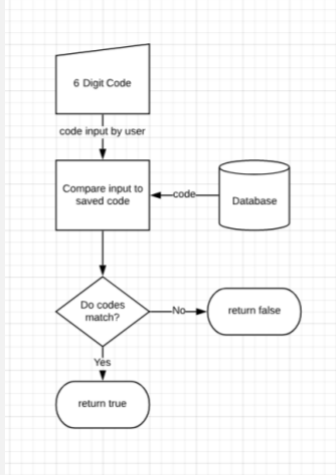
3.3.4.1.1 emailAddress: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Identify user
Description	Email address of user who wants to log in or register

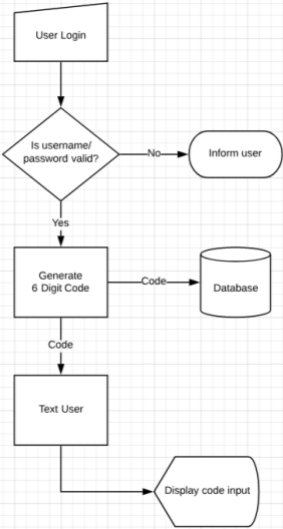
3.3.4.1.2 pinCode: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Code to verify the user has access to their mobile device and can thus prove their identity
Description	6-digit number (code)

3.3.4.1.3 pinIsValid(): boolean

ITEM	DESCRIPTION
Type	Function
Purpose	Verifies the pin
Description	<p>Verifies that the 6-digit number that the user supplies matches the 6-digit number that the app originally generated.</p>  <pre>graph TD; A[/6 Digit Code/] -- "code input by user" --> B[Compare input to saved code]; C[(Database)] -- "code" --> B; B --> D{Do codes match?}; D -- No --> E([return false]); D -- Yes --> F([return true]);</pre> <p>The flowchart illustrates the logic of the pinIsValid() function. It begins with a parallelogram representing the input of a '6 Digit Code' from the user. This input is then processed in a rectangle labeled 'Compare input to saved code'. A cylinder representing the 'Database' provides a 'code' to the same comparison step. Following the comparison, a decision diamond asks 'Do codes match?'. If the answer is 'No', the flow leads to an oval labeled 'return false'. If the answer is 'Yes', the flow leads to an oval labeled 'return true'.</p>
Parameters	None
Returns	Returns true if the numbers match and false if they do not
Functions Called	None

3.3.4.1.4 startProcess(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Starts the Two Factor Authentication process
Description	<p>The application generates the 6-digit number, sends it to the user's mobile phone and then the user needs to type in the same number in order to prove their identity.</p>  <pre> graph TD A[User Login] --> B{Is username/ password valid?} B -- No --> C([Inform user]) B -- Yes --> D[Generate 6 Digit Code] D -- Code --> E[(Database)] D -- Code --> F[Text User] F --> G[/Display code input/] </pre>
Parameters	None
Returns	None
Functions Called	None

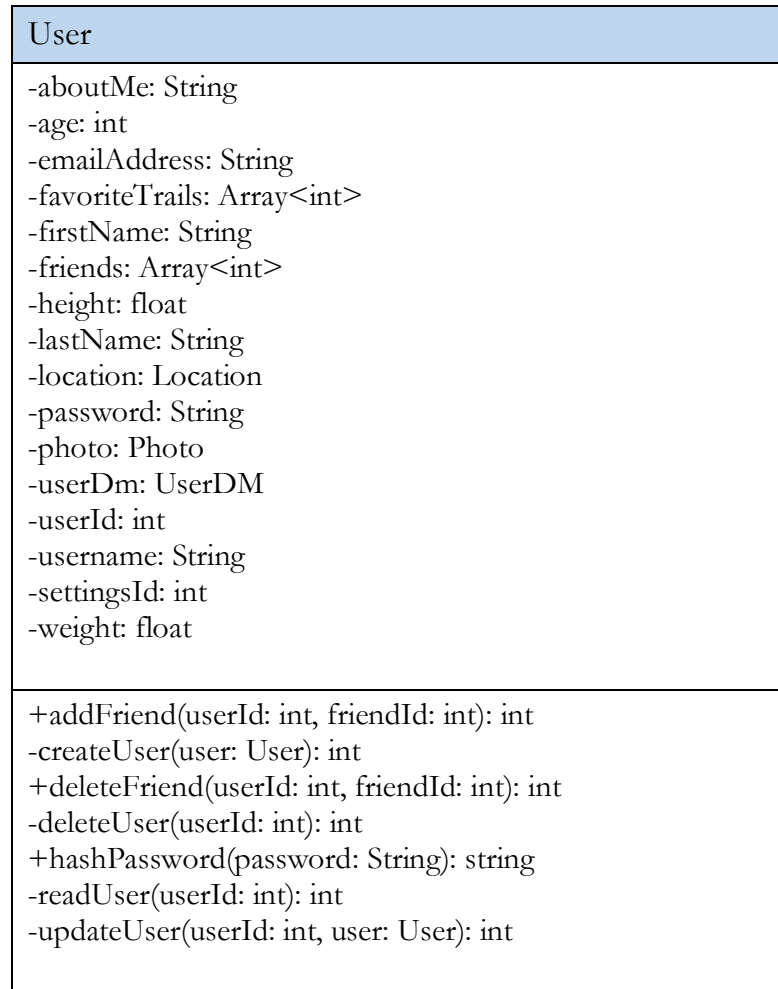
3.3.4.1.5 Design Concerns

SRS	Content
3.2.2.2.2.1	Provided a registered email address, a Two-Factor Authentication email is sent to the User with a 6-digit numeric code, which the screen will prompt for.

3.3.5 User

Description: A user of the application. Friends, non-friends, and the current user are all examples of Users. Contains a list of all hikes the User has been on. Contains data such as name, their location data, a unique identifier, login information.

Class Diagram



3.3.5.1 Design Elements

3.3.5.1.1 aboutMe: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Message where the user describes his or herself.
Description	Users will be able to briefly describe themselves and their interests in up to 300 characters.

3.3.5.1.2 age: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Keep track of the age entered by the user.
Description	The age will be used by the health features to help calculate calories burned.

3.3.5.1.3 emailAddress: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Address allowing the user to be notified of in-app events.
Description	When events happen inside the app, e-mails can be generated to notify the user.

3.3.5.1.4 favoriteTrails: Array<int>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Keep track of trails the user would like to revisit.
Description	This will keep a list of references to the user's favorite trails.

3.3.5.1.5 firstName: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores the first name of the user.
Description	Represents the first name of the user.

3.3.5.1.6 friends: Array<int>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Keep track of the user's friends.
Description	This will keep a list of references to the user's friends.

3.3.5.1.7 height: float

ITEM	DESCRIPTION
Type	Attribute
Purpose	Keep track of the height entered by the user.
Description	The height will be used by the health features to help calculate calories burned.

3.3.5.1.8 lastName: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores the last name of the user.
Description	Represents the last name of the user.

3.3.5.1.9 location: Location

ITEM	DESCRIPTION
Type	Attribute
Purpose	Location used for identifying nearby trails, and for plotting a user's position while on a hike.
Description	Latitude & Longitude coordinates will be available to other app functions to localize the user's experience.

3.3.5.1.10 password: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows the user to access their account securely.
Description	A String of characters between 8-32 characters in length, using a capital, a number, and one special character.

3.3.5.1.11 photo: Photo

ITEM	DESCRIPTION
Type	Attribute
Purpose	Unique picture to allow the user to personalize their account.
Description	Users will be able to upload their own photos as an icon for their profile or use the default image.

3.3.5.1.12 userDm: UserDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store user data management object
Description	This method will talk to database to retrieve, update, and delete information about user object

3.3.5.1.13 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	A unique identifier, which can never be changed, used by the database to identify a specific user.
Description	The userId will be used by the system to uniquely identify the user. This will be different from the displayName, which may have duplicates.

3.3.5.1.14 username: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Unique identifier for the user to use when logging into the app.
Description	This name will only be visible to the user. This is different from the display name which will be visible to other users.

3.3.5.1.15 weight: float

ITEM	DESCRIPTION
Type	Attribute
Purpose	Keep track of the weight entered by the user.
Description	The weight will be used by the health features to help calculate calories burned.

3.3.5.1.16 addFriend(userId: int, friendId: int): int

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Creates a friend relationship between two Users.
Description	Relies upon the userDm class to create the friend relationship between the two Users. Takes userId and friendId as parameters.
Parameters	userId: int, friendId: int
Returns	The return value for the SQL code
Functions Called	userDm.createFriend(userId: int, friendId: int): int

3.3.5.1.17 createUser(user: User): int

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Takes a User and adds it to the database
Description	Relies upon the userDm class to add the User to the database. user is passed as a parameter. Takes user as a parameter.
Parameters	user: User
Returns	The return value for the SQL code
Functions Called	userDm.create(user: User): int

3.3.5.1.18 deleteFriend(userId: int, friendId: int): int

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Takes a userId and friendId and removes the friendship relationship between them.
Description	Relies upon the userDm class to remove the friendship relationship between two users. Takes userId and friendId as parameters.
Parameters	userId: int, friendId: int
Returns	The return value for the SQL code execution
Functions Called	userDm.deleteFriend(userId: int, friendId: int): int

3.3.5.1.19 deleteUser(userId: int): int

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Takes a userId and deletes the associated User from the database.
Description	Relies upon the userDm class to delete the specified User. Takes userId as a parameter. Takes userId as a parameter.
Parameters	userId: int
Returns	The return value for the SQL code
Functions Called	userDm.delete(userId: int): int

3.3.5.1.20 hashPassword(password: String): String

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Takes a password from the user and creates a hash from it so it can later be sent to the database for more secure storage
Description	The user's password is hashed into an encrypted form for storage in the database. <pre>HashPassword(password) { hashedPassword = sha256(); hashedPassword.update(password); return hashedPassword.hexidigest(); }</pre>
Parameters	password: String
Returns	String
Functions Called	None

3.3.5.1.21 readUser(userId: int): int

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Takes a userId and reads in the User object
Description	Relies upon the userDm class to read all of the values from the specified userId row in the User table to populate this object with data. Takes userId as a parameter.
Parameters	userId: int
Returns	The return value for the SQL code execution
Functions Called	userDm.read(userId: int): int

3.3.5.1.22 updateUser(userId: int, user: User): int

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Replace the user data in the database with a new User object.
Description	Relies upon the userDm class to update the User data for the user specified by the userID. Takes userId and user as parameters.
Parameters	userId: int, user: User
Returns	The return value for the SQL code execution
Functions Called	userDm.update(userId: int, user: User): int

3.3.5.1.23 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.3.2.3	The Current Hike tab of the My Hikes displays a Current Hike Card. The card is updated in real time using the User's age, weight, and height in the User's Profile screen to produce a number of calories the User has burned.
3.2.3.3	Tapping the "Stop" button on a Current Hike card Current Hike Card stops updating the number of calories.
3.2.3.10.1	User's progression on the hike is based off of the user's starting location.
3.2.6.5	User's Profile screen contains a profile picture and 300-character autobiography.
3.2.6.5.6.1	Tapping the "Friends" button on the Profile screen navigates the User to the Friends screen.

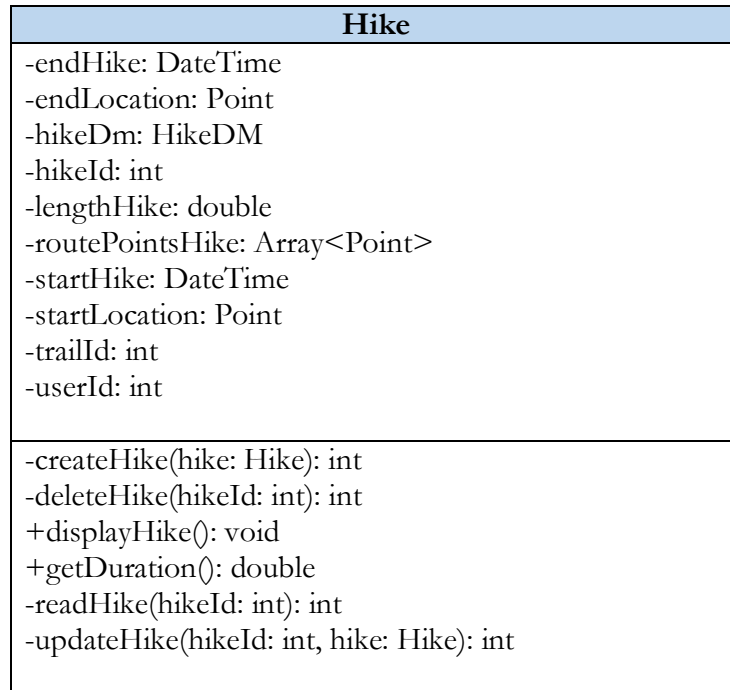
3.4 Exercise

This view contains information related to the user's hiking activity and encouragement to help them maintain a healthy exercise regimen.

3.4.1 Hike

Description: The hike class provides the functions used to manage, maintain and view a currently taken hike or a previously taken hike tied to a specific user in the application.

Class Diagram



3.4.1.1 Design Elements

3.4.1.1.1 endHike: DateTime

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores the DateTime when the Hike was finished.
Description	A variable to contain the exact time and date when the hike was finished.

3.4.1.1.2 endLocation: Point

ITEM	DESCRIPTION
Type	Attribute
Purpose	Identifies the point location where the user ends.
Description	A point location in the database that identifies the x and y coordinates of the users end location.

3.4.1.1.3 hikeDm: HikeDM

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Provides access to the Hike table in the database
Description	The data management class for Hike

3.4.1.1.4 hikeId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Identifies a unique hike in the database
Description	A unique place in the database and a distinct number to use for sorting and search operations in the application.

3.4.1.1.5 lengthHike: double

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The length, measured in feet, of the hike the user walked from start to finish.
Description	The length is calculated from recorded data regarding the parent trail. Can be translated into other units of measurement for display purposes according to the user settings and the length itself (if the number of feet is over 5280, it will be displayed in miles instead of feet)

3.4.1.1.6 routePointsHike: Array<Point>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A reference to all geographic coordinate points that make up the hike
Description	Holds latitude and longitude coordinates of each point of the hike. The points create the route of the hike when connected in order.

3.4.1.1.7 startHike: DateTime

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Stores the DateTime when the Hike was begun.
Description	A variable to contain the exact time and date when the hike was begun.

3.4.1.1.8 startLocation: Point

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Identifies the point location where the user starts
Description	A point in the database that identifies the x and y coordinates of the users start location.

3.4.1.1.9 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Stores the trail's id
Description	A variable to point to a specific trail.

3.4.1.1.10 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Stores the user's id
Description	A variable to store the user's id.

3.4.1.1.11 createHike(hike: Hike): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Creates a hike for the user
Description	Create a new instance of a hike in the database using the provided hike object.
Parameters	hike: Hike
Returns	Returns the SQL return code
Functions Called	hikeDm.create(hike: Hike)

3.4.1.1.12 deleteHike(hikeId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Deletes a hike from the Hike table
Description	Deletes a hike from the Hike table using the provided hikeId
Parameters	hikeId: int
Returns	The SQL return code
Functions Called	hikeDm.delete(hikeId: int):int

3.4.1.1.13 displayHike(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Renders and draws all information pertaining to the hike on a card so the user may see it all conveniently.
Description	Displays the Hike's information as a UI Card, including the duration, the points the route went through, and the elapsed duration. <pre>{ numDrawnPoints = 0 WHILE Var numDrawnPoints LESS THAN size of coordinate-array { DRAW coordiante point AT routePointHike array - POSITION numDrawnPoints numDrawnPoints + 1 } }</pre>
Parameters	None
Returns	None
Functions Called	getDuration()

3.4.1.1.14 getDuration(): double

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Calculates the time it took the user to hike from start to finish
Description	Returns a double representing the time it took the user to hike the trail, measured in seconds. duration = finishTimeHike - startTimeHike
Parameters	None
Returns	double
Functions Called	None

3.4.1.1.15 readHike(hikeId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads a hike from the Hike table
Description	Reads a hike from the Hike table using the provided hikeId
Parameters	hikeId: int
Returns	The SQL return code
Functions Called	hikeDm.read(hikeId: int):int hikeDm.getHike(): Hike

3.4.1.1.16 updateHike(hikeId: int, hike: Hike): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Updates a hike in the Hike table
Description	Updates a hike in the Hike table using the provided hikeId and hike object
Parameters	hikeId: int, hike: Hike
Returns	The SQL return code
Functions Called	hikeDm.update(hikeId: int):int

3.4.1.1.17 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Provide a “My Hikes” screen.
3.2.3.2.2	A Current Hike Card displays the User’s step count, speed (in MPH or KPH) and hike path.
3.2.3.2.4	Functionality is provided to a Current Hike card’s “Stop” and “Save” buttons.
3.2.3.3	Tapping the “Stop” button on a Current Hike Card halts updates to User step count, speed, and hike path.
3.2.3.4	Upon tapping the “Stop” button, a Modal appears in the Current Hike tab of the My Hikes screen prompting the User to rate the hike from 1 star to 5 stars
3.2.3.5	Tapping the “stop” button produces a summary containing the User’s average speed, total step count, and image of the recorded path is provided.
3.2.3.6	Tapping the “Save” on a Current Hike Card causes the card to disappear.
3.2.3.6.1	Information is provided to the History tab to display cards containing a summary of each completed hike. The summary contains a user’s average speed, step count and recorded path.
3.2.3.7	Tapping the Image Card-Button of the user’s hike path navigates the User to the Map screen and display the hike path.

3.4.2 Stopwatch

Description: Supports the UI for keeping track of how long a User has been on a hike.

Class Diagram

Stopwatch
-pairQuantity: int -timeSequencePairs: List<Tuple(Datetime, Datetime)>
+endStopwatch(): void +getTimeElapsed(): float +startStopwatch(): void

3.4.2.1 Design Elements

3.4.2.1.1 pairQuantity: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Keep a reference for the current number of pairs in timeSequencePairs
Description	An integer that contains the current number of datetime start/finish pairs for the stopwatch

3.4.2.1.2 timeSequencePairs: List<Tuple(Datetime, Datetime)>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains all starts and stops for this instance of the stopwatch
Description	A mutable list of tuples that contains a start datetime and a finish datetime

3.4.2.1.3 endStopwatch(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Adds a datetime to the second part of the current timeSequencePair
Description	References the current pair by using the count of pairQuantity as the index to update the current datetime in the second element of the tuple in the array timeSequencePairs. Then adds one to the pairQuantity. Like this: timeSequencePairs[pairQuantity][1].updateTime(currentTime) pairQuantity += 1
Parameters	None
Returns	Void
Functions Called	None

3.4.2.1.4 getTimeElapsed(): float

ITEM	DESCRIPTION
Type	Function
Purpose	Returns an up-to-date hike duration so the controller can populate the UI element that tells the user how long they have been hiking
Description	Returns an up-to-date hike duration that represents how long the user has been on a certain hike by subtracting the start from the finish for each of the values in the timeSequencePairs array.
Parameters	None
Returns	Returns a float that represents the number of seconds that the user has been on the hike
Functions Called	None

3.4.2.1.5 startStopwatch(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Adds a new element to the list timeSequencePairs
Description	Adds a new Tuple(datetime, datetime) to the timeSequencePairs List, adding the current time to the first element of the Tuple, like so: timeSequencePairs[pairQuantity][0].updateTime(currentTime)
Parameters	None
Returns	Returns a float that represents the number of seconds that the user has been on the hike
Functions Called	None

3.4.2.1.6 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.3.2.2	A Current Hike Card contains a stopwatch timer incrementing every millisecond. The formatting is as: 00:00:00 (minutes, seconds, milliseconds)
3.2.3.3	Tapping the “Stop” button on a Current Hike Card will stop the stopwatch timer.
3.2.3.5	The stopwatch time will be provided in the summary created after tapping the “Stop” button the Current Hike Card.

3.5 Trail Information

This view contains all information necessary to display trails and their ancillary views.

3.5.1 AdditionalInfoController

Description: A controller that contains additional information pertaining to the trail such as fees, opening and closing times.

Class Diagram

AdditionalInfoController
-additionalInfoDm: AdditionalInfoDM -hoursOfOperation: String -entranceFee: float -parkingRegulations: String
+display(): void +feeTotal(): float -readAdditionalInfo(trailId: int): int

3.5.1.1 Design Elements

3.5.1.1.1 additionalInfoDm: AdditionalInfoDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.5.1.1.2 hoursOfOperation: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Informs user of the park or trails time of closing
Description	String containing the time of closing

3.5.1.1.3 entranceFee: float

ITEM	DESCRIPTION
Type	Attribute
Purpose	Informs user of the entrance fee
Description	Float containing the cost of entering the trail or park

3.5.1.1.4 parkingRegulations: String

ITEM	DESCRIPTION
Type	Attribute

Purpose	Informs user of the parking fees
Description	Float containing the cost of parking

3.5.1.1.5 feeTotal(): float

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Sums the total cost
Description	Sum of the total cost to park and enter the park or trail <pre>{ float total = entranceFee + parkingFee; return total; }</pre>
Parameters	None
Returns	The total fee
Functions Called	None

3.5.1.1.6 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Displays the information
Description	Updates the view with data from the AdditionalInfoController class.
Parameters	None
Returns	None
Functions Called	None

3.5.1.1.7 readAdditionalInfo(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads additional information from database
Description	Retrieves information about: <ul style="list-style-type: none"> • hoursOfOperation: String • entranceFee: float • parkingRegulations: String • additionalFees: float Passes the trailId to retrieve the correct row
Parameters	trailId: int
Returns	The return SQL code
Functions Called	additionalInfoDM.read(trailId: int)

3.5.1.1.8 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Provide an Additional Information screen.
3.2.4.1	Tapping an additional information button navigates the User to the associated additional information screen.
3.2.4.1.1	Information for fees, season-specific hours, and parking regulations are provided to the Additional Information screen.

3.5.2 Amenity

Description: Object container for Amenity objects.

Class Diagram

Amenity
-amenityId: int -amenityName: String -amenityType: int
N/A

3.5.2.1 Design Elements

3.5.2.1.1 amenityId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the database ID for the amenity.
Description	An integer that represents the amenity in the Amenity table.

3.5.2.1.2 amenityName: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the name of the amenity.
Description	A String that contains the name for this amenity.

3.5.2.1.3 amenityType: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the type of Amenity this is for properly displaying the icon for it on the map.
Description	An integer that represents the type of amenity this Amenity is.

3.5.3 AmenitiesController

Description: Controller for the Amenities Screen

Class Diagram

AmenitiesController
-amenities: Array<Amenity> -amenitiesTrailDm: AmenitiesTrailDM -trailId: int
+display(): void +readAmenities(trailId): int

3.5.3.1 Design Elements

3.5.3.1.1 amenities: Array<Amenity>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the Amenity objects for the view to be able to display and access.
Description	Amenity objects that have been fetched from the database through the read() method.

3.5.3.1.2 amenitiesTrailDm: AmenitiesTrailDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.5.3.1.3 trailId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The ID of the trail from which this view associates its data.
Description	The integer Trail ID from which this view will get its data.

3.5.3.1.4 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Displays amenity information
Description	Updates the view with data from the AmenitiesController class.
Parameters	None – acts on internal class properties
Returns	None
Functions Called	None

3.5.3.1.5 readAmenities(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Displays amenity information
Description	Fetches data from the AmenitiesDM class and uses setters to assign this attributes to Amenity class.
Parameters	trailId: int
Returns	The SQL return code (integer)
Functions Called	AmenitiesDM.read(trailId: int) AmenitiesDM.getAmenities()

3.5.3.1.6 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Provide an Amenities screen.
3.2.4.11	Navigate to the Amenities screen from the trail description screen.
3.2.4.11.1	Display a list of amenities on the amenities screen.

3.5.4 BrowseController

A controller that contains a list of trails populated by the TrailSuggestionGenerator.

Class Diagram

BrowseController
-generator: TrailSuggestionGenerator -trails: Array<Trail>
+display(): void +read(generator.getResults(): Array<int>): int

3.5.4.1 Design Elements

3.5.4.1.1 generator: TrailSuggestionGenerator

ITEM	DESCRIPTION
Type	Attribute
Purpose	The object finds the trails that user has not found yet
Description	This object is used to generate new trails for a user that the user has not hiked yet.

3.5.4.1.2 trails: Array<Trail>

ITEM	DESCRIPTION
Type	Attribute
Purpose	The trail contains a list of trails that will be browsed in this controller's view.
Description	Contains Trail objects

3.5.4.1.3 display(): void

ITEM	DESCRIPTION
Type	Function
Purpose	This method renders and displays the Trails in the view.
Description	This method renders and displays the Trails in the view.
Parameters	N/A
Returns	N/A
Functions Called	N/A

3.5.4.1.4 read(generator.results(): Array<int>): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	The method retrieves all of the Trails using the IDs provided by the generator
Description	The method retrieves all of the Trails using the IDs provided by the generator
Parameters	generator.results(): Array<int>
Returns	The SQL return code
Functions Called	generator.results(): Array<int> trails.foreach(read(trailId: int)): int

3.5.4.1.5 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Provide a Browse screen.
3.2.1.3	Tapping the “Browse” icon on the Bottom Navigation Panel displays the Browse screen.

3.5.5 Comment

Description: A container class for Comment objects.

Class Diagram

Comment
-comment: String -commentId: int -commentThreadId: int -location: Point -userId: int
N/A

3.5.5.1 Design Elements

3.5.5.1.1 comment: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the text data of the comment itself.
Description	Contains a String that is to be displayed as the text content of the comment itself.

3.5.5.1.2 commentId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the id of the comment itself
Description	Contains an integer Comment ID that references the comment in the database

3.5.5.1.3 commentThreadId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the id of the parent Thread.
Description	Contains an integer CommentThreadID that references the comment thread that spawned/owns this comment.

3.5.5.1.4 location: Point

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows the Data Management layer to be accessed by this class.
Description	This Array holds all of the Comments in this Thread.

3.5.5.1.5 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows the Data Management layer to be accessed by this class.
Description	This Array holds all of the Comments in this Thread.

3.5.5.1.6 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.4.12.1	A new comment's text is submitted by the "leave comment" button
3.2.4.12.1.1	Upon comment submission, the app navigates the user back to leave comment screen. The comment input is cleared
3.2.4.12.1.2	Users can rate comments left by other users, as well as comment on other users' comments
3.2.4.12.1.2.1	Users can edit or delete any comment or rating they submitted
3.2.4.12.1.2.2	Tapping the delete comment button will navigate the user back to the Trail Description screen
3.2.4.12.1.2.3	Tapping the "X" icon in the header of the leave comment screen will take the user to the Trail Description screen
3.2.12.1.2.4	Tapping the checkmark icon in the Leave Comment header will navigate the user to the Trail Description screen
3.2.4.13	A means to rate comments and thus determine their validity is provided
3.2.4.13.1	A comment is reported by tapping the "Report comment" button
3.2.6.6.2.1	Comments contain a longitude and latitude coordinates
3.2.6.6.3	Comments can be posted to the community Comment Thread by tapping on the Fab Button at the bottom right corner of the Community Forum screen
3.2.6.6.3.1	Tapping the Fab Button navigates the user to the Drop-Pin Map screen
3.2.6.6.3.3	Tapping the "Create Post" button navigates the User to the Community Forum screen with the Comment field auto populated with a picture of the dropped pin location
3.2.6.6.3.3.1	The User may add text to the Comment text field
3.2.6.6.3.3.2	The post button posts the comment when tapped
3.2.6.6.5	Comments on the community chat Thread display the posted date
3.2.6.6.2.1	Each post includes longitude and latitude coordinates with some User-inputted text

3.5.6 CommentThreadController

Description: A comment thread is a tree of comments which starts with one main comment or question and groups the responses and sub-responses. Comment threads are not restricted to groups of users. In the SRS this controller is also be called the Community Forum.

Class Diagram

CommentThreadController
-comments: Array<Comment> -commentThreadDm: CommentThreadDM -commentThreadId: int -trailId: int
+create(comment: Comment): int +delete(commentId: int): int +display(): void +read(trailId: int): int +update(commentId: int, comment: Comment): int

3.5.6.1 Design Elements

3.5.6.1.1 comments: Array<Comment>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Holds a list of Comments to populate the Comment Thread with.
Description	This Array holds all of the Comments in this Thread.

3.5.6.1.2 commentThreadDm: CommentThreadDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.5.6.1.3 commentThreadId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains this comment thread's ID
Description	An integer that represents the database key for this comment thread.

3.5.6.1.4 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains this comment thread's parent trail ID
Description	An integer that represents the database key for the trail for this comment thread.

3.5.6.1.5 create(comment: Comment): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function allows a new Comment to be added to the Thread.
Description	This function adds the provided Comment to the end of the list, updating the database with the new Comment.
Parameters	Takes the Comment to be added to the thread.
Returns	The SQL code for the success of the operation.
Functions Called	commentThreadDm.createComment(commentThreadId: int, comment: Comment)

3.5.6.1.6 delete(commentId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function allows a specified Comment ID to be removed.
Description	Calls the delete method of the CommentThreadDM object. If the commentId exists in the list of comments, it is allowed to be deleted, otherwise this method fails.
Parameters	Takes an int indicating the ID of the Comment to be removed.
Returns	An integer denoting the success of the SQL command.
Functions Called	commentThreadDm.deleteComment(commentThreadId: int, commentId: int)

3.5.6.1.7 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function displays the comments in the view.
Description	Displays each of the comments chronologically in the view.
Parameters	N/A
Returns	void
Functions Called	N/A

3.5.6.1.8 read(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function updates the comments array with new data from the database.
Description	This function updates the comments array with data from the database.
Parameters	trailId: int
Returns	The return SQL code
Functions Called	commentThreadDM.read(trailId: int): int

3.5.6.1.9 update(commentId: int, comment: Comment): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function allows a specified Comment ID to be removed.
Description	Calls the delete method of the CommentThreadDM object. If the commentId exists in the list of comments, it is allowed to be deleted, otherwise this method fails.
Parameters	Takes an int indicating the ID of the Comment to be removed.
Returns	An integer denoting the success of the SQL command.
Functions Called	commentThreadDm.updateComment(commentId: int, comment: Comment)

3.5.6.1.10 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.4.12	A comment system is provided on the trail description screen
3.2.12.1.2.5	New comments display at the top of the comment thread. Comments made by the user show in a different color
3.2.4.12.2	Show a list of the first 6 comments with a “Read more” button below the list
3.2.4.12.2.1	When tapping “Read More”, the app fetches the next 6 comments and appends them to the list
3.2.4.12.3	Users may post a comment on the comment thread by tapping the “Post” button
3.2.4.12.4	When tapping the “Post” button, the app displays a read-only version of the post at the top of the comments list
3.2.4.12.5	The app indicates if there are no comments for a trail
3.2.6.6	Upon tapping the “Community Forum” button in the Trail Description screen, the app navigates the User to the Community Forum
3.2.6.6.1	Upon first-time navigation to the Community Forum screen, a pop-up Modal appears with a message indicating how the forum is to be used
3.2.6.6.2	The Community Forum screen consists of a limited community Comment Thread
3.2.6.6.4	Comments on the community chat Thread are visible to all Users
3.2.6.6.1	When the user navigates to the Community Forum screen for the first time, a pop-up Modal appears with a message indicating how the forum is to be used
3.2.6.6.2	The Community Forum screen consists of a limited community Comment thread
3.2.6.6.3	The User posts comments to the community Comment thread by tapping on the Fab Button on the Community Forum screen
3.2.6.6.3.3	Tapping the “Create Post” button shall navigate the User to the Community Forum screen with the Comment field auto populated with a picture of the dropped pin’s location
3.2.6.6.3.3.1	The User may add text to the Comment text field
3.2.6.6.3.3.2	The User may post Comments by tapping on the Post button
3.2.6.6.3.4	In the read-only Community Forum screen, the User taps on the picture of the dropped pin location in a post
3.2.6.6.4	Comments on the community chat thread shall be visible to all Users
3.2.6.6.5	Comments on the community chat thread shall display the posted date

3.5.7 EmergencyContactInfoController

Description: A controller that contains information pertaining to contacting emergency resources.
(park ranger name and phone #)

Class Diagram

EmergencyContactInfoController
-rangerDm: RangerDM -rangerId: int -rangerName: String -rangerPhoneNumber: String
+display(): void -readData(rangerId: int): int

3.5.7.1 Design Elements

3.5.7.1.1 rangerDm: RangerDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.5.7.1.2 rangerId: int

Item	Description
Type	Attribute
Purpose	Represents the ID of the ranger who takes care of the area around this trail.
Description	Holds the ID of the ranger in the database.

3.5.7.1.3 rangerName: String

Item	Description
Type	Attribute
Purpose	Represents the name of the ranger who's phone number is also listed.
Description	Used to store the ranger's name and display when needed.

3.5.7.1.4 rangerPhoneNumber: String

<i>Item</i>	<i>Description</i>
Type	Attribute
Purpose	Represents the name of the ranger's phone number.
Description	Used to store the ranger's phone number and display when needed.

3.5.7.1.5 display(): void

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Display all emergency contact info
Description	Updates the view with the park ranger's name and phone number.
Parameters	None
Returns	None
Functions Called	None

3.5.7.1.6 readData(rangerId: int): int

<i>Item</i>	<i>Description</i>
Type	Function
Purpose	Retrieves the data from the database to return the proper information associated with this trail's emergency contact info.
Description	Uses rangerId to retrieve corresponding information from the database. Retrieves information about: <ul style="list-style-type: none">• firstName: String• lastName: String• phoneNumber: int
Parameters	rangerId:int
Returns	The return SQL code
Functions Called	rangerDm.read(rangerId: int)

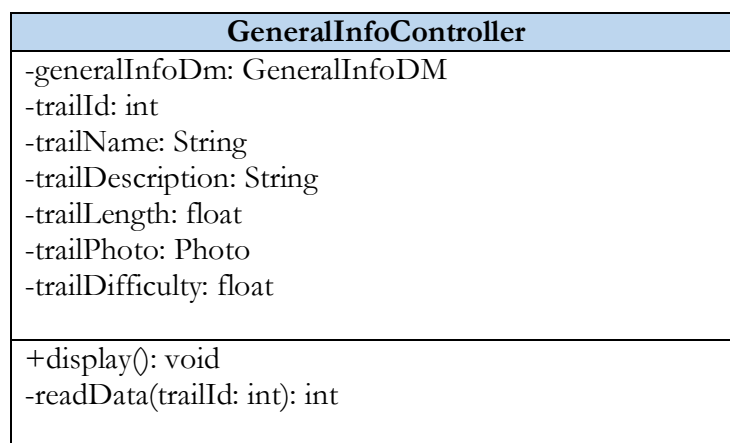
3.5.7.1.7 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	The app has an Emergency Contact Information screen
3.2.4.7	The User navigates to the Emergency Contact Information screen by tapping a button
3.2.4.7.1 3.2.4.7.1.1 3.2.4.7.1.2	The Emergency Contact information screen shows the park ranger's name and phone number

3.5.8 GeneralInfoController

Description: Handles the basic information associated with a trail including description, length, etc.

Class Diagram



3.5.8.1 Design Elements

3.5.8.1.1 generalInfoDm: GeneralInfoDM

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.5.8.1.2 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The trail's identifier
Description	Holds the trail's id in the database

3.5.8.1.3 trailName: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Identify trail for user
Description	A text name corresponding to the trail, by which it is known to users.

3.5.8.1.4 trailDescription: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Describe attributes of a trail for a user.
Description	A textual depiction of a trail that will help user's determine a hike's qualities and decide which trail to hike.

3.5.8.1.5 trailLength: float

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The trail's length in km or miles.
Description	A number corresponding to the length of the trail.

3.5.8.1.6 trailPhoto: Photo

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Show the user what a trail looks like visually.
Description	A photo that contains features of the hike.

3.5.8.1.7 trailDifficulty: float

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Communicate difficulty level of a hike.
Description	A number scale that will allow users to pick hikes relevant to their abilities.

3.5.8.1.8 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Uses the object's data to update the view.
Description	Updates the view with UI elements including: <ul style="list-style-type: none">• Comment Thread• Trail Description• Trail Photo
Parameters	None
Returns	None
Functions Called	None

3.5.8.1.9 readData(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Retrieves the data from the database to return the proper information associated with the trail
Description	Uses rangerId to retrieve corresponding information from the database. Retrieves information about: <ul style="list-style-type: none">• trailName: String• trailDescription: String• trailLength: float• trailPhoto: Photo• trailDifficulty: float
Parameters	trailId: int
Returns	The return SQL code
Functions Called	generalInfoDM.read(trailId: int)

3.5.8.1.10 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.9.2	Each Trail Description screen has a General Information section
3.2.1.7.1	The Settings screen has a General section
3.2.3.1.3	The User may toggle notifications on the app's Settings screen in the General section
3.2.4.3.1	The trail length defaults to miles or km depending on the user's location
3.2.4.3.2	The user has the option to toggle between miles and kilometers
3.2.6.2.1	The app navigates the User to the People picker screen where the User may select one or more than one person

3.5.9 HikingEquipment

Description: Object container for HikingEquipment objects.

Class Diagram

HikingEquipment
-hikingEquipmentId: int -hikingEquipmentPhoto: Photo -hikingEquipmentDescription: string
N/A

3.5.9.1 Design Elements

3.5.9.1.1 hikingEquipmentId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the database ID for the hiking equipment.
Description	An integer that represents the hiking equipment in the Hiking_Equipment table.

3.5.9.1.2 hikingEquipmentPhoto: Photo

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains a Photo for the hiking equipment.
Description	A photo of the hiking equipment, retrieved from the database.

3.5.9.1.3 hikingEquipmentDescription: string

ITEM	DESCRIPTION
Type	Attribute
Purpose	Describes this piece of hiking equipment.
Description	A string that contains a detailed description of what kind of hiking equipment would serve the user well for this trail.

3.5.10 Plant

Description: Object container for Plant objects.

Class Diagram

Plant
-plantDescription: string -plantId: int -plantIsDangerous: string -plantIsEdible: string -plantPhoto: Photo
N/A

3.5.10.1 Design Elements

3.5.10.1.1 plantDescription: string

ITEM	DESCRIPTION
Type	Attribute
Purpose	Describes this species of plant.
Description	A string that contains a detailed description a plant specie that might be found along the trail.

3.5.10.1.2 plantId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the database ID for the plant.
Description	An integer that represents the plant in the Plant table.

3.5.10.1.3 plantIsDangerous: string

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains a string that describes if the plant is dangerous to people or not.
Description	Contains a string that describes if the plant is dangerous to people or not.

3.5.10.1.4 plantIsEdible: string

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains information about the edibility of the plant.
Description	Contains a string that describes if the plant is edible or not.

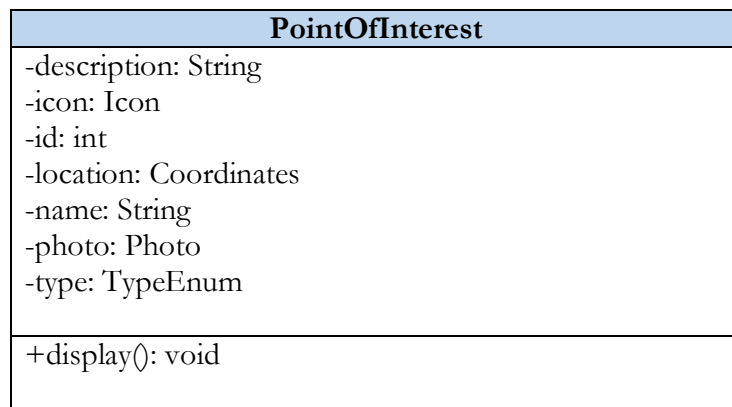
3.5.10.1.5 plantPhoto: Photo

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains a Photo of the plant.
Description	A photo of the plant, retrieved from the database.

3.5.11 PointOfInterest

Description: This class contains all the relevant information for a point of interest. A point of interest is a location, feature, or service seen along the trail.

Class Diagram



3.5.11.1 Design Elements

3.5.11.1.1 description: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	To help the user identify what the point of interest is about
Description	A String that acts as a description

3.5.11.1.2 icon: Icon

ITEM	DESCRIPTION
Type	Attribute
Purpose	For ease of use
Description	A visual to represent the point of interest

3.5.11.1.3 id: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	To help the system identify the point of interest
Description	An integer to represent the point of interest. Unique to each point of interest within the trail.

3.5.11.1.4 location: Coordinates

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	To accurately place the point of interest on the map
Description	A couple floats to indicate where the point of interest is located on a map.

3.5.11.1.5 name: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A String for the identification and reference of the Point of Interest
Description	A String that acts as the name

3.5.11.1.6 photo: Photo

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	These images provide a comprehensive view of the focal point and its purpose.
Description	An unordered array of photos

3.5.11.1.7 type: TypeEnum

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	To allow the program to tell which Icon to associate with this Point of Interest
Description	An enumeration that acts as a type for the Point of Interest. Possible values: <ul style="list-style-type: none">• Location• Feature• Service

3.5.11.1.8 display(): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	So that the user can quickly and easily view the point of interest information
Description	Updates the view with data from the PointOfInterest class. This includes the name, icon, description, and photos for the point of interest.
Parameters	Parameters go here
Returns	Return values go here
Functions Called	Any other methods called directly by this method are listed here

3.5.11.1.9 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	The app has a Points of Interest screen
3.2.4.10	The app takes the user to the points of interest screen after tapping the points of interest button in the trail description screen
3.2.4.10.1	The points of interest screen displays: <ul style="list-style-type: none">● Point of interest name● Photo of the point of interest● Details behind the point of interest

3.5.12 PointsOfInterestController

A controller that handles points of interest for a trail. This controller is tied into the map and will contain information and photos pertaining to each point of interest.

Class Diagram

PointsOfInterestController
-pointOfInterestDm: PointOfInterestDM -pointsOfInterest: Array<PointOfInterest> -trailId: int
+display(): void -readData(trailId: int): int

3.5.12.1 Design Elements

3.5.12.1.1 pointOfInterestDm: PointOfInterestDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer
Description	The Data Management object for this class. Provides methods for accessing the database

3.5.12.1.2 pointsOfInterest: Array<PointOfInterest>

ITEM	DESCRIPTION
Type	Attribute
Purpose	A place to store all the points of interest for a given trail
Description	An array of points of interest for the trail

3.5.12.1.3 trailId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores the trailId that corresponds to the points of interest that the trail has.
Description	An integer trail ID.

3.5.12.1.4 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Displays the points of interest on the map
Description	The function finds points of interest from the pointsOfInterest array and places them on the map
Parameters	None
Returns	None
Functions Called	None

3.5.12.1.5 readData(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Retrieves the data from the database to return the points of interest
Description	Uses a trailId to retrieve corresponding information from the database. Retrieves information about: <ul style="list-style-type: none">• pointsOfInterest: List<PointOfInterest>
Parameters	pointsOfInterestId:int
Returns	The return SQL code
Functions Called	pointOfInterestDM.read(trailId: int) pointOfInterestDM.getPointsOfInterest()

3.5.12.1.6 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	The app has a Points of Interest screen
3.2.4.10	The app navigates to the points of interest screen after tapping the points of interest button
3.2.4.10.1	The points of interest screen displays the following: <ul style="list-style-type: none">• Point of interest name• Photo of the point of interest• Details behind the point of interest

3.5.13 Review

A container class for Review objects.

Class Diagram

Review
-comment: string -userId: int -rating: int -reviewId: int -trailId: int
N/A

3.5.13.1 Design Elements

3.5.13.1.1 comment: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the text data of the comment itself.
Description	Contains a String that is to be displayed as the text content of the comment itself.

3.5.13.1.2 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the ID of the user who made this review
Description	Contains an integer foreign key id of the user who created this review.

3.5.13.1.3 rating: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the trail rating for this review
Description	Contains an integer that represents the rating of the trail that the user attached to this review, must be an integer, 1 – 5.

3.5.13.1.4 reviewId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the primary key review ID
Description	Contains an integer value for the review's ID/primary key.

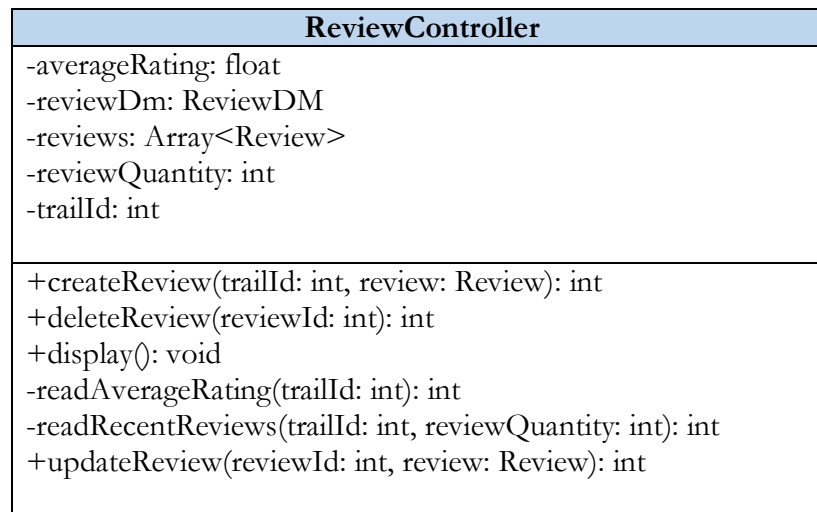
3.5.13.1.5 trailId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the ID of the trail that this review is attached to
Description	Contains an integer that represents the foreign key ID of the trail that this review is reviewing.

3.5.14 ReviewController

A controller that displays the most recent reviews for a given trail, allowing users to post new reviews as necessary. Also aggregates the reviews to provide an overall score to the trail itself.

Class Diagram



3.5.14.1 Design Elements

3.5.14.1.1 averageRating: float

ITEM	DESCRIPTION
Type	Attribute
Purpose	A value that contains the aggregated review scores of all reviews for this trail
Description	A float value that is calculated from the database using all of the reviews for the associated trail

3.5.14.1.2 reviewDm: ReviewDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	The data management object enables this class to read/write to the database.
Description	The data management object enables this class to read/write to the database.

3.5.14.1.3 reviews: Array<Review>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the most recent reviews that have been posted to this trail
Description	Contains an array of Review objects that represent the reviews that this controller displays in its view

3.5.14.1.4 reviewQuantity: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the current quantity of loaded reviews in memory
Description	Contains an integer representing how many reviews this controller currently has loaded.

3.5.14.1.5 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the ID of the trail associated with these reviews
Description	Contains an integer foreign key to the trail that these reviews are associated with

3.5.14.1.6 createReview(review: Review): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Creates a new review for this trail
Description	Given a review object, passes it along to the reviewDm object for it to be added to the table
Parameters	review: Review
Returns	An integer code denoting success or failure of retrieval
Functions Called	reviewDm.createReview(review: Review): int

3.5.14.1.7 deleteReview(reviewId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Disassociates the review from the trail.
Description	Given a reviewId, relies upon the reviewDm object to disassociate the review at that reviewId row from its trailId, deleting the trailId from that row.
Parameters	reviewId: int
Returns	An integer code denoting success or failure of retrieval
Functions Called	reviewDm.deleteReview(reviewId: int): int

3.5.14.1.8 display(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Displays the objects in this object to the view
Description	Displays the objects in this object to the view
Parameters	N/A
Returns	N/A
Functions Called	N/A

3.5.14.1.9 readAverageRating(trailId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Retrieves the average of all reviews for the given trail
Description	Relies upon the reviewDm class to aggregate all of the review ratings for the provided trailId together into one float value. The float value must be within 1 – 5 or it is invalid. Puts the collected float into the averageRating variable.
Parameters	trailId: int
Returns	An integer code denoting success or failure of retrieval
Functions Called	reviewDm.readAverageRating(trailId: int): int reviewDm.getAverageRating(): float

3.5.14.1.10 readRecentReviews(trailId: int, reviewQuantity: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Retrieves Review objects from at most the 6 most recent reviews posted to this trail. If this method is called sequentially, it will add 6 to the reviewQuantity.
Description	<p>Relies upon the reviewDm class to gather at most the reviewQuantity + 6 most chronologically recent reviews from the review table given the trailId specified. If there are less than reviewQuantity + 6 reviews, it will collect them all.</p> <p>Puts the collected Reviews into the reviews array.</p> <p>Adds 6 to the value stored in reviewQuantity.</p>
Parameters	trailId: int
Returns	An integer code denoting success or failure of retrieval
Functions Called	reviewDm.readRecentReviews(trailId: int, reviewQuantity: int): int reviewDm.getReviews(): Array<Review>

3.5.14.1.11 updateReview(reviewId: int, review: Review): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Updates the provided reviewId with the provided review.
Description	Relies upon the reviewDm class to replace the review at row reviewId with the values provided in the passed review object.
Parameters	reviewId: int, review: Review
Returns	An integer code denoting success or failure of retrieval
Functions Called	reviewDm.updateReview(reviewId: int, review: Review)

3.5.15 SafetyGuidelinesController

A controller that contains information pertaining to trail safety, including four different subsections: Predatory Wildlife, Edible Plants, Dangerous Plants, and Hiking Equipment.

Class Diagram

SafetyGuidelinesController
-hikingEquipment: Array<HikingEquipment> -hikingEquipmentDm: HikingEquipmentDM -plantDm: PlantDM -plants: Array<Plant> -trailId: int -wildlife: Array<Wildlife> -wildlifeDm: WildlifeDM
+display(): void +readSafetyGuidelines(trailId: int): int

3.5.15.1 Design Elements

3.5.15.1.1 hikingEquipment: Array<HikingEquipment>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the data pertaining to many HikingEquipment objects
Description	Contains an array of HikingEquipment objects retrieved from the database

3.5.15.1.2 hikingEquipmentDm: HikingEquipmentDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer
Description	The Data Management object for this class. Provides methods for accessing the database.

3.5.15.1.3 plantDm: PlantDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer
Description	The Data Management object for this class. Provides methods for accessing the database

3.5.15.1.4 plants: Array<Plant>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the data pertaining to many Plant objects
Description	Contains an array of Plant objects retrieved from the database.

3.5.15.1.5 trailId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	A Trail ID that references the Trail that is connected to this Controller.
Description	A String Trail ID that can be used to access Firebase for data access.

3.5.15.1.6 wildlife: Array<Wildlife>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the data pertaining to many Wildlife objects
Description	Contains an array of Wildlife objects retrieved from the database.

3.5.15.1.7 wildlifeDm: WildlifeDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.5.15.1.8 display(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Displays the information
Description	Returns the array containing the traversal data
Parameters	N/A
Returns	An array of TraversalInfo objects
Functions Called	N/A

3.5.15.1.9 readSafetyGuidelines(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Retrieves HikingEquipment, Plant, and Wildlife objects from their respective tables in the database.
Description	Given a trailId int, queries each of the hikingEquipmentDm, plantDm, and wildlifeDm classes to retrieve their respective objects.
Parameters	trailId: int
Returns	An integer code denoting success or failure of retrieval
Functions Called	hikingEquipmentDm.read(trailId: int) hikingEquipmentDm.getHikingEquipment() plantDm.read(trailId: int) plantDm.getPlants() wildlifeDm.read(trailId: int) wildlifeDm.getWildlife()

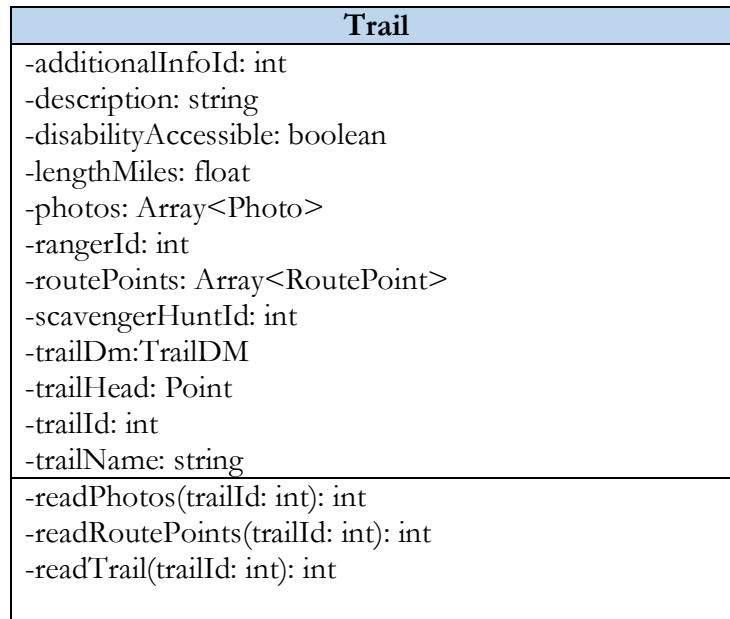
3.5.15.1.10 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	The app has a Safety Guidelines screen
3.2.4.8	Tapping the “Safety Guidelines” button will navigate the user to the Safety Guidelines screen
3.2.4.8.1	The Predatory Wildlife, Dangerous Plant Life, Edible Plant Life, and Hiking Equipment sections are included in the safety guidelines
3.2.4.8.2	The predatory wildlife section shows a photo of each predator, a description of the predator, and a description of the predator warning signs
3.2.4.8.3	The edible plant life section shows a photo of each plant, and a description of the plant
3.2.4.8.4	The dangerous plant life section shows a photo and description of each plant
3.2.4.8.5	The hiking section shows a photo and description of each hiking equipment
3.2.4.9	The app alerts the user about hunting locations on the Trail

3.5.16 Trail

Description: A geographical path with a predefined start and finish that is in a natural setting of some kind. Contains all data for an individual trail including rating, photos, where the trailhead is.

Class Diagram



3.5.16.1 Design Elements

3.5.16.1.1 additionalInfoId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to stored additional info id integer
Description	An additional info id which refers to specific additional information about unique trail

3.5.16.1.2 description: string

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store description of the trail object
Description	A description string that provides information about trail

3.5.16.1.3 disabilityAccessible: boolean

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store boolean value if the trail is disability accessible
Description	A boolean value that is stored in the database to represent on Trail Description Controller if current trail is disability accessible or not

3.5.16.1.4 lengthMiles: float

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store miles length as a float
Description	A float value that represents length of the trail in miles

3.5.16.1.5 photos: Array<Photo>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store array of Photo objects
Description	An array of photo objects that can be viewed on the trail

3.5.16.1.6 rangerId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used for storing ranger id
Description	A ranger id which identifies unique ranger in the table to contact for emergency information and help

3.5.16.1.7 routePoints: Array<RoutePoint>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store array of route points in the object
Description	Stores route points in the trail for users to see and use

3.5.16.1.8 scavengerHuntId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store scavenger hunt it integer
Description	A scavenger hunt id which identifies scavenger hunt location in the database

3.5.16.1.9 trailDm: TrailDM

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store trail data management object to set attributes with
Description	A trail data management is used to store data management object to retrieve data from database

3.5.16.1.10 trailHead: Point

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used for storing the location of trail head
Description	A location object which stored coordinates of the trail head

3.5.16.1.11 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used for storing trail ID in the object
Description	An integer that identifies unique trail in the app.

3.5.16.1.12 trailName: string

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Used to store trail name in the object
Description	A trail name which identifies a trail in the app when it is shown to the user

3.5.16.1.13 readPhotos(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads the photos in to enable their display
Description	This method calls trailDm.readPhotos() to retrieve all of the photo files associated with this Trail for their eventual display.
Parameters	trailId: int
Returns	The return SQL code
Functions Called	trailDm.readPhotos(trailId: int)

3.5.16.1.14 readRoutePoints(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads trail information from database
Description	Retrieves routePoints from the trailDm object.
Parameters	trailId: int
Returns	The return SQL code
Functions Called	trailDm.readRoutePoints(trailId: int)

3.5.16.1.15 readTrail(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads trail information from database
Description	<p>Retrieves the following data:</p> <ul style="list-style-type: none">• trail_head: Point• rangerId: int• map_id: int• additional_info_id: int• scavenger_hunt_id: int• trail_name: String• description: String• disability_accessible_yn: bool• num_steps: int• length_miles: float <p>Also calls readPhotos() in order to load the photos into memory.</p>
Parameters	trailId: int
Returns	The return SQL code
Functions Called	trailDm.read(trailId: int) readPhotos(trailId: int)

3.5.16.1.16 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.3.4.1	When tapping any star on the Modal, the User is taken to that Trail's Comment screen where the selected star rating is auto populated
3.2.3.9	Store the approximate number of steps taken to finish the trail
3.2.4.3	The app displays the trail's length under the General Information section

3.5.17 TrailDescriptionController

Description: The controller which handles getting the trail's additional information and routing to views.

Class Diagram

TrailDescriptionController
-reviewController: ReviewController -trailId: int -trail: Trail
-getWeatherData(): Weather -goToAdditionalInformation(trailId: int): void -goToAmenities(trailId: int): void -goToCommunityForum(trailId: int): void -goToEmergencyContact(id: int): void -goToSafetyGuidelines(trailId: int): void -goToPointsOfInterest(trailId: int): void -goToScavengerHunt(trail.getScavengerHuntId(): int): void -startHike(userId: int, trailId: int): int -favoriteTrail(): int -goToTrailMap(): void -readTrail(trailId: int): int

3.5.17.1 Design Elements

3.5.17.1.1 trail: Trail

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store trail object in the controller
Description	The trail stores information of all its attributes, which are accessed from the controller.

3.5.17.1.2 trailId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store trail id in the controller
Description	Stores trail id in the controller so it can be easily passed to other controller to retrieve information from the database

3.5.17.1.3 reviewController: ReviewController

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to hold review controller object
Description	Stores reviewController object so it can be displayed on TrailDescription controller

3.5.17.1.4 getWeatherData(): Weather

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This gets the weather forecast for the trail.
Description	The weather API will fetch the weather for the trail here.
Parameters	N/A
Returns	Weather
Functions Called	Weather API getWeather()

3.5.17.1.5 goToAdditionalInformation(trail.getAdditionalInformationId(): int): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Go to the AdditionalInfoController
Description	This will take the user to the trail's additional information. It is passing trail id as a parameter
Parameters	trailId: int
Returns	Void
Functions Called	trail.getAdditionalInformationId()

3.5.17.1.6 goToAmenities(trailId: int): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Go to the AmenitiesController
Description	This will take the user to the trail's amenities view. It is passing trail id as a parameter
Parameters	trailId: int
Returns	Void
Functions Called	HomeController.openView(AmenitiesController(trailId))

3.5.17.1.7 goToCommunityForum(trailId: int): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Take the user to the hike's community forum view.
Description	Navigate the user to the community forum from a button click.
Parameters	trailId: int
Returns	Void
Functions Called	HomeController.openView(CommentThreadController(trailId))

3.5.17.1.8 goToEmergencyContact(trail.getRangerId()): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Take the user to the emergency contact view.
Description	This will take the user from a button click to the trail's emergency contact information view. Passes the integer rangerId to the EmergencyContactInfoController.
Parameters	trail.getRangerId()
Returns	Void
Functions Called	HomeController.openView(EmergencyContactInfoController(rangerId))

3.5.17.1.9 goToSafetyGuidelines(trailId: int): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Take the user to the safety guidelines view.
Description	This will route the user from a button click to the safety guidelines view, passing an integer trailId along.
Parameters	trailId: int
Returns	Void
Functions Called	HomeController.openView(SafetyGuidelinesController(rangerId))

3.5.17.1.10 goToPointsOfInterest(trailId: int): void

ITEM	DESCRIPTION
Type	Function
Purpose	Take the user to the hike's points of interest area
Description	This will route the user from a button click to the trail's points of interest view. It is passing trail id as a parameter
Parameters	trailId: int
Returns	Void
Functions Called	HomeController.openView(PointsOfInterestController(trailId: int))

3.5.17.1.11 goToScavengerHunt(trail.getScavengerHuntId(): int): void

ITEM	DESCRIPTION
Type	Function
Purpose	If a scavenger hunt exists for a given trail, take the user to that page.
Description	The user will click a button and will be navigated to the scavenger hunt area of the trail. If trail.getScavengerHuntId() == NULL: this method will do nothing else HomeController.openView(ScavengerHuntController(trail.getScavengerHuntId()))
Parameters	trail.getScavengerHuntId(): int
Returns	void
Functions Called	trail.getScavengerHuntId(): int HomeController.openView(ScavengerHuntController(trail.getScavengerHuntId()))

3.5.17.1.12 startHike() : void

ITEM	DESCRIPTION
Type	Function
Purpose	Take the user to the main hike page.
Description	After the user starts the hike, they will be taken to the hike page, which starts the step count, and saves the user's location. <pre>{ stepCount.start(); location.save(); trail.toHome(); }</pre>
Parameters	N/A
Returns	void
Functions Called	N/A

3.5.17.1.13 favoriteTrail(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	The user will click a star icon to indicate that the trail is one of their “favorites”.
Description	After the trail is saved as one of the user’s favorites, the icon turns into a filled icon. <pre>{ SET favorite to TRUE; }</pre>
Parameters	N/A
Returns	void
Functions Called	N/A

3.5.17.1.14 goToTrailMap(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Take the user to the google maps map of the trail.
Description	This takes the user to the google map of the trail. This will happen when the user clicks on the button to view the trail’s map.
Parameters	N/A
Returns	void
Functions Called	Google Maps API view map

3.5.17.1.15 readTrail(trailId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Fetch the trail data given the id
Description	This method uses TrailDM to fetch all appropriate trail data from the database given an integer trailId
Parameters	trailId: int
Returns	Success code
Functions Called	trail.readTrail(id: int)

3.5.17.1.16 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	The app has a Trail Description screen
3.2.1.8.1	The Browse screen has a Suggested Trails section
3.2.1.10	Each Trail Description screen has buttons for Additional Information, Emergency Contact Information, Safety Guidelines, Points of Interest
3.2.3.2.1	Upon tapping the “Start” button on the Trail Description screen, the app navigates the User to the Current Hike tab of the My Hikes screen
3.2.5.11	The app displays a “Directions” button on the Trail Description screen
3.2.5.11.1	When tapping the “Directions” button on the Trail Description screen, the Google Maps app opens to provide directions for the User to the trailhead
3.2.5.11.2	When tapping the “Directions” button on the Trail Description screen and the User does not have Google Maps installed, a Modal appears indicating that Google Maps must be installed to use this feature
3.2.6.6	Upon tapping the “Community Forum” button in the Trail Description screen, the app takes the User to the Community Forum screen
3.2.6.7	On trails with an easy and moderate Difficulty rating, the app displays a “Scavenger Hunt” button in the Trail Description screen

3.5.18 TrailSuggestionGenerator

Description: A class that determines nearby trails to suggest to the user. This uses algorithms that prioritizes trails that are rated well, that have not been hiked before by the user, and that are relatively near in proximity.

Class Diagram

TrailSuggestionGenerator
-candidateTrails: Array<Trail> -hikeDm: HikeDM -recentHikes: Array<Hike> -user: User -userId: int
-compareNearbyTrails(): void -getRecentTrails(): Array<int> -readData(userId: int): int -results(): Array<int> -sortByRating(): void

3.5.18.1 Design Elements

3.5.18.1.1 candidateTrails: Array<Trail>

ITEM	DESCRIPTION
Type	Attribute
Purpose	This array stores all of the trail suggestions.
Description	As the functions of the class are called, a list of trails is added to this variable, and edited until only viable trail suggestions remain.

3.5.18.1.2 recentHikes: Array<Hike>

ITEM	DESCRIPTION
Type	Attribute
Purpose	An array of the user's recent hikes
Description	An array of hikes that the user has been on recently

3.5.18.1.3 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	The User object that provides the user's favorite trails
Description	The User object that is fetched through the userId

3.5.18.1.4 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The ID of this user
Description	The integer ID of this user

3.5.18.1.5 compareNearbyTrails(): void

ITEM	DESCRIPTION
Type	Function
Purpose	This functions helps limit the recommendations to trails within a reasonable distance of the User.
Description	<p>This function finds trails whose locations are within the given radius and stores them in a local Trail array. It calls GetRecentTrails(), then compares its own trail array with those returned by GetRecentTrails(). If any of the trails in its own array are also found in the GetRecentTrails() array, it removes the duplicate trail from the local array. It then stores the local array in the candidateTrails variable.</p> <pre>{ trails = GET trails(LOCATION); recentTrails = getRecentTrails(); FOR (int i = 0; i < trails.size(); i++) IF trails[i] != recentTrails[i] candidateTrails.push_back(trails[i]); }</pre>
Parameters	N/A
Returns	N/A
Functions Called	GetRecentTrails()

3.5.18.1.6 getRecentTrails(): Array<int>

ITEM	DESCRIPTION
Type	Function
Purpose	This function helps avoid repeat recommendations.
Description	This function fetches the hiker's trail history, then returns an array of Trail IDs.
Parameters	N/A
Returns	Trail array
Functions Called	N/A

3.5.18.1.7 readData(userId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	This function helps return results of suggested trails
Description	This method fetches all of the recent hikes the user has been on as well as the user's data, using the User object and the HikeDM class.
Parameters	userId: int
Returns	The SQL return code
Functions Called	user.read(userId: int): int hike.readHikesByUser(userId: int): int

3.5.18.1.8 results(): Array<int>

ITEM	DESCRIPTION
Type	Function
Purpose	This function helps return results of suggested trails
Description	<p>This function uses getRecentTrails(), compares with compareNearbyTrails(), and sorts the nearby trails that were not recently hiked by a user. Once that is done the list is sorted by rating using sortByRating(). Then the list is returned.</p> <pre>{ recentTrails = getRecentTrails() nearbyTrails = compareNearbyTrailz() candidateTrails = nearbyTrails - recentTrails sortByRating() RETURN }</pre>
Parameters	N/A
Returns	ArrayList<int>
Functions Called	N/A

3.5.18.1.9 sortByRating(): int

ITEM	DESCRIPTION
Type	Function
Purpose	This function helps users find the best trails by making sure highly-rated trails are suggested first.
Description	<p>This function reads the candidateTrails variable and uses a sorting algorithm to place the highest rated trails at the top of the recommendation list and the lowest rated trails at the bottom. Once it is done sorting it updates candidateTrails with the new, sorted array.</p> <pre> { int n = candidateTrails.num FOR iPivot<-n-1 ... 1 swapped<-false FOR iCheck<-0 ... iPivot-1 IF array[iCheck] > array[iCheck+1] swap array[iCheck] and array[iCheck+1] swapped<-true IF !swapped RETURN } </pre>
Parameters	N/A
Returns	N/A
Functions Called	N/A

3.5.18.1.10 Design Concerns

SRS	Content
3.2.3.1.1	Upon tapping the Push Notification, the app navigates the User to the Browse screen under the “Suggested Trails” section
3.2.3.1.2	If the app has launched prior to the User tapping on the Push Notification, then the app navigates the User to the Browse screen under the “Suggested Trails” section

3.5.19 TraversalInfoController

Description: A controller that contains information pertaining to trail traversal, such as disability access and gradient and elevation change.

Class Diagram

TraversalInfoController
-trailId: int -traversalInfoDm: TraversalInfoDM
+display(): void

3.5.19.1 Design Elements

3.5.19.1.1 trailId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	A Trail ID that references the Trail that is connected to this Controller.
Description	A Trail ID that can be used to access Firebase for data access.

3.5.19.1.2 traversalInfoDm: TraversalInfoDM

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.5.19.1.3 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Displays the information
Description	Updates the UI to display the suggested trails.
Parameters	-
Returns	An array of TraversalInfo objects
Functions Called	-

3.5.19.1.4 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.9.3	Each Trail Description screen has a traversal section
3.2.4.4	The app shows the trail's difficulty in the general information section
3.2.4.4.1	Trail difficulties consist of easy, moderate, and hard
3.2.4.5	The app indicates to the user if the trail is disability-accessible
3.2.4.6	The app shows the trail's change in elevation

3.5.20 Weather

Description: A weather forecast system that has information of what the weather is likely to be for the next hour or few hours. It will be queried from an external database.

Class Diagram

Weather
+forecast: Json
+update(): void +display(): void

3.5.20.1 Design Elements

3.5.20.1.1 forecast: Json

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores the weather info in a format that is easy to read and write to.
Description	A Json file containing weather information

3.5.20.1.2 update(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Keeps the weather information up-to-date
Description	This function retrieves the newest weather information from the cloud using Firebase, then stores the info in the forecast variable update() { firebase.open(); forecast = firebase; }
Parameters	None
Returns	None
Functions Called	None

3.5.20.1.3 display(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Shows the weather information to the user
Description	Updates the UI to display data parsed from the forecast variable.
Parameters	None
Returns	None
Functions Called	None

3.5.20.1.4 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.9.1	Each Trail Description screen has a weather section
3.2.4.14	The app fetches and displays a four-day weather forecast for the trail's area

3.5.21 Wildlife

Description: Object container for Wildlife objects.

Class Diagram

Wildlife
-wildlifeDescription: string -wildlifeId: int -wildlifePhoto: Photo -wildlifeName: string -wildlifeWarningSigns: string
N/A

3.5.21.1 Design Elements

3.5.21.1.1 wildlifeDescription: string

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Describes this species of wildlife.
Description	A string that contains a detailed description of what kind of wildlife the user may find on their hike.

3.5.21.1.2 wildlifeId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the database ID for the wildlife.
Description	An integer that represents the wildlife in the Wildlife table.

3.5.21.1.3 wildlifePhoto: Photo

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains a Photo for the wildlife.
Description	A photo of the wildlife, retrieved from the database.

3.5.21.1.4 wildlifeName: string

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The name for this wildlife.
Description	A string that contains the name for this species of wildlife.

3.5.21.1.5 wildlifeWarningSigns: string

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Describes how and where this wildlife may be found, particularly what to look out for.
Description	A string that contains a detailed description of warning signs warning for the presence of one of these creatures.

3.6 Social

This view contains all controllers and classes necessary to enable the interpersonal social aspects of the application, including friends, location tracking, and messaging.

3.6.1 EditProfileController

Description: A controller that features forms that allow a user to edit their User Profile information.

Class Diagram

EditProfileController
-user: User -userId: int
+readUser(userId: int): int +updateUser(userId: int, user: User): int

3.6.1.1 Design Elements

3.6.1.1.1 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores user object attributes which will be edited
Description	A user of the application. Friends, non-friends, and the current user are all examples of Users. Contains a list of all hikes the User has been on. Contains data such as name, their location data, a unique identifier, login information.

3.6.1.1.2 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store user id integer attribute
Description	Stores used id integer to get data about the profile in the database

3.6.1.1.3 readUser(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to retrieve data for a given user, then store that in the user object.
Description	Calls the read method of the userDM object. If successful, it stores the values returned from reading the database into the user object, then return 1 indicating success. If there was an error reading the information, it returns 0.
Parameters	userId: int
Returns	SQL return code
Functions Called	user.read(userId)

3.6.1.1.4 updateUser(userId: int, user: User): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to update the database to match any updated data in the user object.
Description	Calls the update method of the userDM object, passing in the userID, and the user object itself. If userDM.update() was successful returns 1, this function also returns 1 indicating success. Otherwise it returns 0.
Parameters	userId: int, user: User
Returns	SQL return code
Functions Called	user.update(userId, user)

3.6.2 FavoriteTrailsController

Description: A controller that contains a list of all trails the User has marked as a favorite.

Class Diagram

FavoriteTrailsController
-user: User -userId: int
+display(): void +readUser(userId: int): int

3.6.2.1 Design Elements

3.6.2.1.1 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	Object that holds user specific information.
Description	Object that will hold user specific information and be populated by the information that is entered by the user.

3.6.2.1.2 userID: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Stores a unique ID for a user
Description	Specific and unique integer set to one user.

3.6.2.1.3 display(): void

ITEM	DESCRIPTION
Type	Function
Purpose	This function displays the trails a user has favorited
Description	Updates the UI to display the trails a user has favorited. This will be done by querying the database for the Trail table which will have a column marked if the trail has been favorited by the user. The function will display all rows that have been favorited by the user.
Parameters	None
Returns	None
Functions Called	None

3.6.2.1.4 readUser(userID: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Determines the user based on the given userID
Description	Retrieves the User object from the database:
Parameters	userId: int
Returns	SQL execution code
Functions Called	user.readUser(userId: int): int

3.6.2.1.5 Design Concerns

SRS	Content
3.2.3.8	A trail displays an empty star icon to indicate if the trail is one of the user's favorites
3.2.3.8.1	Trails indicates if it is one of the user's favorite trails
3.2.3.8.1.1	The hikes stores a list of the user's favorite trails. The user can access these from the favorites section

3.6.3 FriendsController

Description: Contains a list of friends the user has connected with, as well as links to their User Profile Controllers.

Class Diagram

FriendsController
-user: User -userId: int
+display(): void +readUser(userId: int): int -goToUserProfileController(userId: int): void

3.6.3.1 Design Elements

3.6.3.1.1 friends: Array<int>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Holds a list of the User ID's to represent the User's friends.
Description	This will allow the FriendsController to display the Friends as a list, fetching their usernames from the database.

3.6.3.1.2 userDm: UserDM

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.6.3.1.3 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Helps organize users as friends to a certain user in the application.
Description	Updates the UI to display a list of friends that are registered as friends in the application. Uses getters of the user object to show list of friends
Parameters	N/A
Returns	N/A
Functions Called	User object getters

3.6.3.1.4 goToUserProfileController(userId: int): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	To navigate to a user profile controller
Description	This function will take us to the user profile controller of a friend
Parameters	userId: int
Returns	None
Functions Called	HomeController.openView(UserProfileController(userId: int)): void

3.6.3.1.5 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Have a friends screen
3.2.6.4.1	When the user taps the vertical ellipses button, a Modal dropdown appears with two buttons: “Invite Friends” and “Remove Friends”
3.2.6.4.1.1	If there is only one person in the thread along with the User, then the “Remove Friends” button is disabled
3.2.6.4.1.3	Upon tapping the “Remove Friends” button, the app navigates the User to a filtered People Picker screen displaying the current members in the thread
3.2.6.4.1.4	Display an info message prompting the User to select the friend(s) to be removed
3.2.6.4.1.5	The User may tap on people to remove and tap on the checkmark to save
3.2.6.4.1.6	Upon selecting people and tapping on the checkmark, the app navigates the User back to the thread, and a message will display indicating that these “people” were removed from the thread
3.2.6.5.6.2	The Friends screen displays the list of friends the User has.
3.2.6.5.6.3	The Friends screen has a button to add friends
3.2.6.5.6.4	Upon Long Pressing a friend in the list, a trash icon appears in the header of the Friends screen
3.2.6.5.6.4.1	Upon tapping the trash icon, the app removes that person from the User’s Friend screen

3.6.4 FriendRequest

Description: A container class for friend requests

Class Diagram

FriendRequest
-friendId: int -friendRequestId: int -message: String -userId: int
N/A

3.6.4.1 Design Elements

3.6.4.1.1 friendId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The user ID of the friend being sent this friend request
Description	The integer user ID of the friend being sent this friend request

3.6.4.1.2 friendRequestId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The ID of this friend request
Description	The integer primary key ID of this friend request

3.6.4.1.3 message: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	An optional string that can be sent with the friend request
Description	An optional string that can be sent with the friend request

3.6.4.1.4 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The user ID of the user who is sending this friend request
Description	The integer user ID of the user who is sending this friend request

3.6.5 FriendRequestController

Description: The controller which handles friend requests. This will handle sending friend requests as well as accepting or declining friend requests between users.

Class Diagram

FriendRequestController
-friendRequestDm: FriendRequestDM -requests: Array<FriendRequest> -user: User -userId: int
-readRequests(userId: int): int -readUser(userId: int): int +resolveFriendRequest(friendRequest: FriendRequest, resolution: boolean): int +sendFriendRequest(friendRequest: FriendRequest): int

3.6.5.1 Design Elements

3.6.5.1.1 friendRequestDm: FriendRequestDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	The DM class that enables interaction with the database on behalf of friend requests
Description	The DM class that enables interaction with the database on behalf of friend requests

3.6.5.1.2 requests: Array<FriendRequest>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains all relevant friend requests to be reviewed by this user.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.6.5.1.3 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	The User object, giving access to its friends array
Description	The User object, giving access to its friends array

3.6.5.1.4 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The ID of this user
Description	The integer primary key of the user

3.6.5.1.5 readRequests(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function reads in the array of friend requests
Description	Uses the userId to read in all relevant friend requests for this user.
Parameters	userId: int
Returns	The SQL return code
Functions Called	friendRequestDm.readRequests(userId: int): int friendRequestDm.getRequests(): Array<FriendRequest>

3.6.5.1.6 readUser(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function retrieves the user object
Description	Uses User's read method to retrieve the User object given the userId.
Parameters	userId: int
Returns	void
Functions Called	user.read(userId: int)

3.6.5.1.7 resolveFriendRequest(friendRequest: FriendRequest, resolution: boolean): int

ITEM	DESCRIPTION
Type	Function
Purpose	This function handles when a user declines a friend request.
Description	If resolution is true: deletes the friend request from the Friend_Request table, and adds the relationship between these two friends in the Friend table. If resolution is false: deletes the friend request from the Friend_Request table, doing nothing else.
Parameters	friendRequest: FriendRequest, resolution: boolean
Returns	The SQL return code
Functions Called	friendRequestDm.delete(friendRequest.getFriendRequestId(): int): int user.addFriend(friendRequest.getFriendId(): int): int

3.6.5.1.8 sendFriendRequest(friendRequest: FriendRequest): int

ITEM	DESCRIPTION
Type	Function
Purpose	This function handles sending the friend requests to specified users.
Description	Creates a new row in the Friend_Request table using the FriendRequest object provided.
Parameters	friendRequest: FriendRequest
Returns	The SQL return code
Functions Called	friendRequestDm.createRequest(friendRequest: FriendRequest): int

3.6.5.1.9 Design Concerns

SRS	Content
3.2.1.1	Have a friends screen.
3.2.6.4.1.2	Upon tapping the “Invite Friends” button, the app navigates the User to the People Picker screen

3.6.6 Location

Description: Accesses and holds the location of the User, updating in real time if the user has enabled that setting.

Class Diagram

Location
-location: Point -locationDm: LocationDM -locationId: int -user: User -userId: int
-createLocation(userId: int, location: Location): int -deleteLocation(userId: int): int -readLocation(userId: int): int -updateLocation(userId: int, location: Location): int

3.6.6.1 Design Elements

3.6.6.1.1 location: Point

ITEM	DESCRIPTION
Type	Attribute
Purpose	Represents the current location of the user in space.
Description	Stores the current location of the user in space.

3.6.6.1.2 locationDm: LocationDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Interact with the database.
Description	This attribute is used to interact with the database.

3.6.6.1.3 locationID: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Unique id for the Location in the Location Table.
Description	Used to identify the Location stored in the Location Table.

3.6.6.1.4 user: User

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the userId for the user.
Description	This contains all the user data obtained from the data.

3.6.6.1.5 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the userId for the user.
Description	Unique integer user ID can be used to retrieve data from User table.

3.6.6.2 createLocation(userId: int, location: Location): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Create a new instance of a location in the database
Description	Creates a new entry in the location table
Parameters	userId: int, location: Location
Returns	The returned code from the LocationDM class
Functions Called	locationDm.create(location: Location)

3.6.6.2.1 delete(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Deletes the row associated to the userId parameter
Description	Deletes entry from the database using userId
Parameters	userId: int
Returns	The SQL return code
Functions Called	locationDm.delete(userId: int): int

3.6.6.2.2 read(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads the location for the userId
Description	Reads the location from the table for the corresponding userId which is passed as a parameter
Parameters	userId: int
Returns	The SQL return code
Functions Called	locationDm.read(userId: int): int

3.6.6.2.3 update(userId: int, location: Location): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Update the location instance.
Description	Updates location object which is passed as a parameter with userId integer
Parameters	userId: int, location: Location
Returns	The SQL return code
Functions Called	locationDm.update(userId: int, location: Location): int

3.6.7 LocationTracker

Description: The location tracker holds the user's coordinates if the user has location tracking toggled on.

Class Diagram

LocationTracker
-user: User -userId: int -userDm: UserDM
+display(): void -readUser(userId: int): int

3.6.7.1 Design Elements

3.6.7.1.1 locationTrackingSettingsDm: LocationTrackingSettingsDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.6.7.1.2 user: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Contains the User's unique String identifier
Description	Allows the program to identify which User is being tracked.

3.6.7.1.3 getLocation(): float, float

ITEM	DESCRIPTION
Type	Function
Purpose	Returns the user's latitude and longitude coordinates.
Description	Accesses the phone's location to enable geographically relative content. if(location tracking enabled) { Point = user.getCurrentLocation() return Point } return null
Returns	Returns two floats

3.6.7.1.4 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.3.10	After the user starts a hike, the app saves the user's current location in the local database.
3.2.6.8	Display a toggle under the General section of the Settings screen that allows the User to enable or disable location tracking with Friends
3.2.6.8.2	When two friends enable location tracking, the User's location appears on the friend's Map screen and the friend's location shall appear on the User's Map screen as profile picture icons
3.2.6.8.3	If the User has location tracking enabled, the User's location is refreshed every 60 seconds
3.2.6.8.4	The User's displayed location is accurate within 5 meters
3.2.6.8.5	If the User's GPS loses its tracking signal, the app records and displays the User's last seen location on the Map screen

3.6.8 Message

Description: A message is text which can be sent to another user (a friend of the user) for social purposes.

Class Diagram

Message
-message: String -messageId: int -messageThreadId: int -userId: int
N/A

3.6.8.1 Design Elements

3.6.8.1.1 message: String

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Contains the text of the message
Description	A string that contains the message text of this message

3.6.8.1.2 messageId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	The primary key of this message
Description	Holds an integer that represents the primary key of this message object

3.6.8.1.3 messageId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The foreign key for the parent message thread
Description	Holds an integer that represents the foreign key for the parent message thread of this message

3.6.8.1.4 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The user who sent this message
Description	Holds an integer that represents the foreign key for the user who created this message.

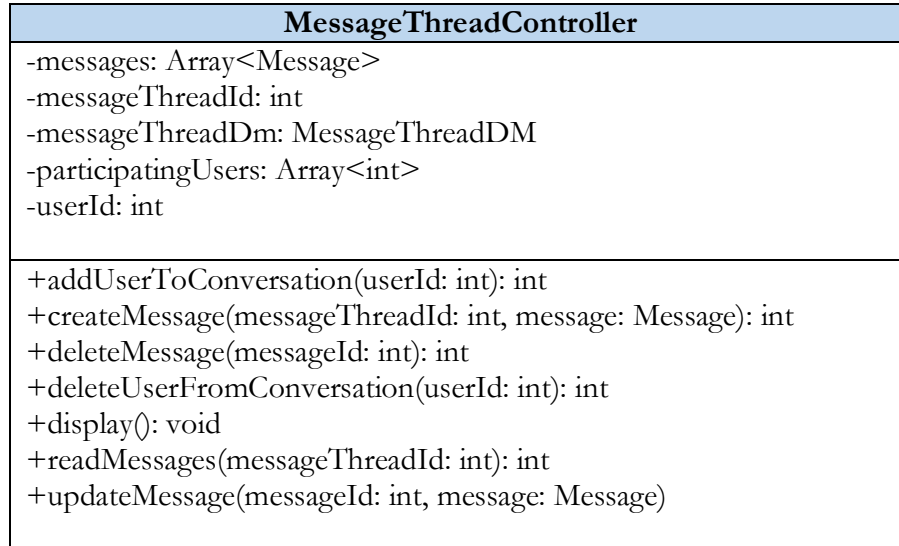
3.6.8.1.5 Design Concerns

SRS	Content
3.2.1.1	Have a Messages screen.
3.2.6.2.2	Tapping the checkmark at the top right of the People Picker screen will save and display the conversation Thread and navigate the User to the Messages screen
3.2.6.3	The User may create a new message by typing (the keyboard will be the default of every system, in other words the same keyboard used when typing any kind of text) in the messages text field at the bottom of each thread, and either tapping on the “Send” icon or tapping the Enter key on the mobile device
3.2.6.3.1	Each message posted will display the read-only version of the message
3.2.6.3.5.3	Only Users who wrote the selected message may delete the selected message

3.6.9 MessageThreadController

Description: A thread is a composition of messages between users. These messages are private between the users in the message group. Related messages in each thread are displayed in the order of creation.

Class Diagram



3.6.9.1.1 Design Elements

3.6.9.1.2 messages: Array<Message>

ITEM	DESCRIPTION
Type	Attribute
Purpose	A message thread is composed of messages in chronological order. This attribute contains those messages.
Description	Contains an ordered list of Message classes.

3.6.9.1.3 messageThreadId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The ID of the message thread in the database
Description	The integer ID of the message thread in the database

3.6.9.1.4 messageThreadDm: MessageThreadDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.6.9.1.5 participatingUsers: Array<int>

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	An Array of integers that represents the IDs of all users who are authorized to view and send messages to this message thread
Description	An Array of integers that represents the IDs of all users who are authorized to view and send messages to this message thread

3.6.9.1.6 userId: int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Indicates the ID of this user
Description	Contains an integer representing a User.

3.6.9.1.7 addUserToConversation(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Adds the provided user to the conversation
Description	Uses messageThreadDm to create a new row in the Message_Thread_Friend table that enables the specified user to see, receive, and send messages to this message thread.
Parameters	userId: int
Returns	The SQL return code
Functions Called	messageThreadDm.createNewThreadRelationship(messageThreadId: int, userId: int)

3.6.9.1.8 createMessage(messageThreadId: int, message: Message): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Adds a new Message to the MessageThread originating from the current User.
Description	Relies upon MessageThreadDM to create a new message for this message thread.
Parameters	Accepts a Message object.
Returns	The SQL return code
Functions Called	messageThreadDm.createMessage(messageThreadId: int, message: Message)

3.6.9.1.9 deleteMessage(messageId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Removes the message from the current message thread
Description	Relies upon the MessageThreadDM to remove the message from the message thread
Parameters	messageId: int
Returns	The SQL return code
Functions Called	messageThreadDm.deleteMessage(messageId: int)

3.6.9.1.10 deleteUserFromConversation(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Removes the user from the current message thread
Description	Relies upon the MessageThreadDM object to update the current thread and remove the specified user from it
Parameters	userId: int
Returns	The SQL return code
Functions Called	messageThreadDm.deleteUserFromThread(messageThread: int, userId: int)

3.6.9.1.11 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Displays the information
Description	Updates the view with data from the MessageThread class.
Parameters	None
Returns	None
Functions Called	None

3.6.9.1.12 readMessages(messageThreadId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Reads all the messages from this message thread
Description	Relies upon the MessageThreadDM object to retrieve all of the messages from this message thread.
Parameters	messageThreadId: int
Returns	The SQL return code
Functions Called	messageThreadDm.readMessagesByThread(messageThreadId: int)

3.6.9.1.13 updateMessage(messageId: int, message: Message): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Updates the specified message
Description	Relies upon the MessageThreadDM to update the specified message with the newer message object provided
Parameters	messageId: int, message: Message
Returns	The SQL return code
Functions Called	messageThreadDm.updateMessage(messageId: int, message: Message)

3.6.9.1.14 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Have a Messages screen.
3.2.1.4	Upon tapping the “Chat” icon on the Bottom Navigation Panel, navigate the User to the Messages screen
3.2.6.1	Provide an Instant Messaging service as a social interactive feature
3.2.6.1.1	Upon tapping the chat icon in the bottom navigation panel, navigate the User to the Messages screen
3.2.6.2	Allow Users to create a new message thread on the Messages screen by tapping on the Fab Button at the bottom right corner
3.2.6.3.5	Upon tapping the vertical ellipses button, a Modal dropdown menu appears displaying an “Edit” button and a “Delete” button
3.2.6.3.5.1	Upon tapping the “Edit” button, the read-only version of the message becomes a text field where the User may edit the comment
3.2.6.3.5.2	Upon tapping the “Delete button, the read-only version of the message is omitted from the thread

3.6.10 MessageThreadsController

Description: A controller which is a parent to MessageThreadController. In this controller user can select a message thread to view.

Class Diagram

MessagesThreadsController
-messageThreadDm: MessageThreadDM -messageThreads: Array<int> -userId: int
+createMessageThread(userId: int, recipientUserId: int): int +deleteMessageThread(messageThreadId: int): int +display(): void -goToMessageThreadController(messageThreadId: int): void +readMessageThreads(userId: int): int

3.6.10.1 Design Elements

3.6.10.1.1 messageThreadDm: MessageThreadDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.6.10.1.2 messageThreads: Array<int>

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.6.10.1.3 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store user id in the controller
Description	Stores user id in the controller to view message threads

3.6.10.1.4 createMessageThread(userId: int, recipientUserId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to add a message thread to the database
Description	Adds a message thread to the database for the corresponding user and recipient user in the database
Parameters	userId: int, recipientUserId: int
Returns	SQL execution code
Functions Called	messageThreadDm.createThread(userId: int, recipientUserId: int): int

3.6.10.1.5 deleteMessageThread(messageThreadId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to delete a message thread to the database
Description	Deletes a message thread from the database
Parameters	messageThreadId: int
Returns	SQL execution code
Functions Called	messageThreadDm.deleteThread(messageThreadId: int): int

3.6.10.1.6 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Used to display message threads in the controller
Description	Displays message threads in the controller using useId: int. Calls read function first to read corresponding message threads for a user. If message threads have been successfully retrieved from the database then it will display them
Parameters	None
Returns	None
Functions Called	None

3.6.10.1.7 goToMessageThreadController(messageThreadId: int): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Adds a new Message to the MessageThread originating from the current User.
Description	Updates the table MessageThreadMessages to add the Message's primary key ID to it. Adds the Message object to the end of the messages list object. messages.add(Message) updateDatabaseTable(MessageThreadMessages, Message)
Parameters	messageThreadId: int
Returns	None
Functions Called	HomeController.openView(messageThreadController(messageThreadId: int)))

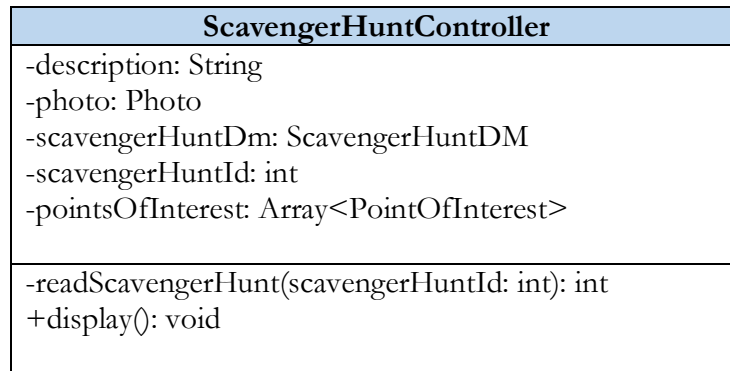
3.6.10.1.8 readMessageThreads(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Adds a new Message to the MessageThread originating from the current User.
Description	Updates the table MessageThreadMessages to add the Message's primary key ID to it. Adds the Message object to the end of the messages list object. messages.add(Message) updateDatabaseTable(MessageThreadMessages, Message)
Parameters	Accepts a Message object.
Returns	None
Functions Called	messageThreadDm.readMessageThreadsByUser(userId: int)

3.6.11 ScavengerHuntController

Description: The Scavenger Hunt Controller contains trail-specific Focal Points which the user may find along the trail. This is a feature found on the Map and is only located on Trails that are rated at lower difficulty ratings.

Class Diagram



3.6.11.1 Design Elements

3.6.11.1.1 description: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	A description that describes the scavenger hunt itself to users
Description	A String that tells the user what to look for during the scavenger hunt.

3.6.11.1.2 photo: Photo

ITEM	DESCRIPTION
Type	Attribute
Purpose	A picture that goes along with the scavenger hunt to help the user know what to look for.
Description	A Photo object that contains a fetched Photo from the database associated with this scavenger hunt.

3.6.11.1.3 scavengerHuntId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The identifier of the given scavenger hunt.
Description	An integer that identifies the scavenger hunt in the database.

3.6.11.1.4 scavengerHuntDm: ScavengerHuntDM

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.6.11.1.5 readScavengerHunt(scavengerHuntId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Retrieves information about Scavenger Hunt from the database and sets all necessary attributes
Description	Updates User Interface to display. Passes scavengerHuntID as a parameter : <ul style="list-style-type: none">• descriptionSH: string• descriptionPI: string• photo: Photo• location: Point
Parameters	scavengerHuntId: int
Returns	The return SQL code
Functions Called	scavengerHuntDM.read(): int

3.6.11.1.6 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Present user with list of items to find
Description	Updates UI to display the list of scavenger hunt items
Parameters	N/A
Returns	N/A
Functions Called	N/A

3.6.11.1.7 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Have a Scavenger Hunt screen.
3.2.6.7	On Trails with an easy and moderate Difficulty rating, the app displays a “Scavenger Hunt” button in the Trail Description screen
3.2.6.7.1	Upon tapping the “Scavenger Hunt” button, the app navigates the User to the Scavenger Hunt screen
3.2.6.7.1	Upon tapping the “Scavenger Hunt” button, the app navigates the User to the Scavenger Hunt screen
3.2.6.7.1.1	The Scavenger Hunt screen contains a list of focal points that can be identified on the trail (Plants, Animals, points of Interest).
3.2.6.7.1.2	Each focal point contains pictures of focal points for easy identification
3.2.6.7.1.3	Each focal point has checkboxes to mark off individual focal points when they are identified

3.6.12 UserProfileController

Description: A controller that displays User data, including a biography, recent hikes, and if the User is considered a Friend to the viewer, their active hiking location.

Class Diagram

UserProfileController
-userId: int -user: User
-display(): void -goToEditProfileController(userId: int): void -readUser(userId: int): int

3.6.12.1 Design Elements

3.6.12.1.1 user: User

ITEM	DESCRIPTION
Type	Attribute
Purpose	Used to store user object in the controller
Description	Stores user object along with all of its attributes to show on the profile controller

3.6.12.1.2 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	A unique identifier, which can never be changed, used by the database to identify a specific user.
Description	The userId will be used by the system to uniquely identify the user. This will be different from the displayName, which may have duplicates.

3.6.12.1.3 goToEditProfileController(userId: int): void

ITEM	DESCRIPTION
Type	Function
Purpose	Go to the EditProfileController
Description	This will take the user to the edit profile controller where user can update their profile information. It is passing user id as a parameter
Parameters	userId: int
Returns	Void
Functions Called	HomeController.openView(EditProfileController(userId: int))

3.6.12.1.4 display(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Displays the information in the view.
Description	Updates the UI to display the user profile information in the view (profilePicture, biography). These are obtained from the user object.
Parameters	None
Returns	None
Functions Called	user.getPhoto(): Photo user.getAboutMe(): string

3.6.12.1.5 readUser(userId: int): int

ITEM	DESCRIPTION
Type	Function
Purpose	Retrieves user attributes from the database
Description	Retrieves user attributes from the database from User table using userDM object by passing user id as a parameter.
Parameters	userId: int
Returns	The SQL return code
Functions Called	user.read(userId: int): int

3.6.12.1.6 Design Concerns

SRS	Content
3.2.6.5.1	If the User opts out of uploading a profile picture, the app saves the default avatar determined by the system (placeholder image) as the User's profile picture
3.2.6.5.2	The Profile screen is accessed by tapping on the User's profile icon at the top right of each screen
3.2.6.5.3	The Profile screen displays the User's name at the top of the screen
3.2.6.5.4.1.1	Upon tapping the checkmark in the Edit Profile screen and uploading a picture (from your local storage, or from a URL), the app navigates the User back to the Profile screen and displays the picture the User set
3.2.6.5.5	The Profile screen displays a read-only version of a 300-character autobiography under the Profile screen's About Me section
3.2.6.5.5.1	The About Me section displays a vertical ellipses button in the top right corner of the section
3.2.6.5.6	The Profile screen provides a "Friends" button displaying the String "Friends"

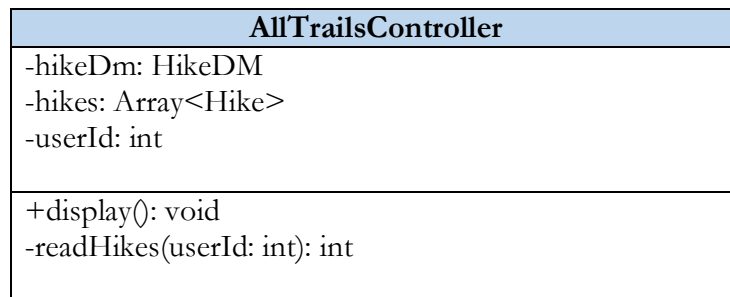
3.7 Map

This view contains all classes and controllers that relate to the displaying and support of interactive maps.

3.7.1 AllTrailsController

Description: A controller that displays a complete history of each hike (as a History Card), displays history cards in descending order by date.

Class Diagram



3.7.1.1 Design Elements

3.7.1.1.1 hikeDm: HikeDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	The HikeDM object that enables access to the database to fetch a list of recent hikes
Description	The HikeDM object that enables access to the database to fetch a list of recent hikes

3.7.1.1.2 hikes: Array<Hike>

ITEM	DESCRIPTION
Type	Attribute
Purpose	The Hike fields used to populate the view with Hikes.
Description	An array of the most recent Hike objects that the user has been on chronologically.

3.7.1.1.3 userId: int

ITEM	DESCRIPTION
Type	Attribute
Purpose	The integer User ID for the Hikes to be fetched.
Description	The integer User ID to allow the Hikes to be fetched from the Database.

3.7.1.1.4 display(): void

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function displays a history of the User's Hikes
Description	This function update the UI to display a list of the hikes a user has gone on. This will be done by querying the database for the table associated to the entity Hikes. This table records a user's hike with a datetime of start, a datetime of finish, and an array of coordinate points that represents the path they took. This data will be displayed to the user in descending order from datetime of finish.
Parameters	User_ID for the database
Returns	None
Functions Called	FirebaseAccess.

3.7.1.1.5 readHikes(userId: int): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	This function retrieves hikes from the database
Description	Uses HikeDM to retrieve Hike objects from the database
Parameters	userId: int
Returns	The return SQL code
Functions Called	hikeDm.readHikesByUser(userId: int): int

3.7.1.1.6 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Have an All Trails screen.
3.2.3.6.3	The History tab of the My Hikes screen displays each History Card in a descending order by date
3.2.3.6.4	The History tab of the My Hikes screen displays up to 10 History Cards
3.2.3.6.4.1	If the User has hiked more than 10 hikes, the History tab of the My Hikes screen displays a “See more” button under the list of History Cards.
3.2.3.6.4.2	Upon tapping the “See more” button (see 3.2.3.6.5.1) on the History tab of the My Hikes screen, the app navigates the User to the All Trails screen.
3.2.3.6.5	The All Trails screen displays a complete history of each hike the User went on (displayed as History Cards) and a “Sort” icon in the screen header
3.2.3.6.5.1	The History Cards are displayed in a descending order by date by default
3.2.3.6.5.2	Upon tapping the “Sort” icon in the All Trails screen, the displays a dropdown Modal with Date, Trail name, Difficulty, and State
3.2.3.6.5.2.1	Upon tapping any selection in the dropdown Modal, the dropdown Modal disappears, and the app navigates the User to the All Trails screen
3.2.3.6.5.2.2	Upon tapping the “Date” selection from 3.2.3.6.7.2, the All Trails screen displays each History Card in a descending order by date
3.2.3.6.5.2.3	Upon tapping the “Trail name” selection from 3.2.3.6.7.2, the All Trails screen displays each History Card in an alphabetical order by Trail name
3.2.3.6.5.2.4	Upon tapping the “Difficulty” selection from 3.2.3.6.7.2, the All Trails screen displays each History Card in an alphabetical order by Trail name that is grouped by Trail Difficulty, starting with the easy Difficulty
3.2.3.6.5.2.5	Upon tapping the “State” selection from 3.2.3.6.7.2, the All Trails screen displays each History Card in an alphabetical order by Trail name that is grouped by state name in alphabetical order

3.7.2 FilterController

Description: Controller for the filter part of the Filter screen

Class Diagram

FilterController
-location: Point -map: Map -matches: Array<Trail> -overlookIsRequired: bool -trailDm: TrailDM -waterfallIsRequired: bool
+filterInfo(): void -readTrailsByFilter(location: Point, overlookIsRequired: boolean, waterfallIsRequired: boolean): int

3.7.2.1 Design Elements

3.7.2.1.1 location: Point

ITEM	DESCRIPTION
Type	Private variable
Purpose	Holds the user's current search coordinates for their Trail search. Defaults to the User's current physical location, but can be specified in the view. Generally, a user will want to search for trails in a specific area.
Description	Location GPS Coordinates for searching.

3.7.2.1.2 map: Map

ITEM	DESCRIPTION
Type	Private variable
Purpose	Directs the user to the Map screen
Description	Submitting filter data will direct the user to the map screen with filter icons overlaying the map screen

3.7.2.1.3 matches: Array<Trail>

ITEM	DESCRIPTION
Type	Private variable
Purpose	Holds an array of Trails for successful matches from the database.
Description	This will allow preliminary information about the Trails to be fetched and displayed as the user specifies search criteria.

3.7.2.1.4 overlookIsRequired: bool

ITEM	DESCRIPTION
Type	Private variable
Purpose	Represents the scenic overlook filter selection
Description	This represents whether or not the user wants to only see trails that have scenic overlooks.

3.7.2.1.5 trailDm: TrailDM

ITEM	DESCRIPTION
Type	Attribute
Purpose	Allows this controller to gain access to the Data Management Layer.
Description	The Data Management object for this class. Provides methods for accessing the database.

3.7.2.1.6 waterfallIsRequired: bool

ITEM	DESCRIPTION
Type	Private variable
Purpose	Represents the waterfall filter selection
Description	This represents whether or not the user wants to only see trails that have waterfalls.

3.7.2.1.7 filterInfo(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Display the filtered user data
Description	It will update the UI to display the data on the map window. It will only show trails that have waterfalls and scenic overlooks if those attributes are set to true. Updates the matches array with Trail ID's that fit in with the local search criteria.
Parameters	None
Returns	None
Functions Called	Firebase Access

3.7.2.1.8 readTrailsByFilter(location: Point, overlookIsRequired: boolean, waterfallIsRequired: boolean): int

<i>ITEM</i>	<i>DESCRIPTION</i>
Type	Function
Purpose	Display the filtered user data
Description	It will update the UI to display the data on the map window. It will only show trails that have waterfalls and scenic overlooks if those attributes are set to true. Updates the matches array with Trail ID's that fit in with the local search criteria.
Parameters	location: Point, overlookIsRequired: boolean, waterfallIsRequired: boolean
Returns	None
Functions Called	trailDm.readTrailsByFilter(location: Point, overlookIsRequired: boolean, waterfallIsRequired: boolean): int trailDm.getTrails(): Array<Trail>

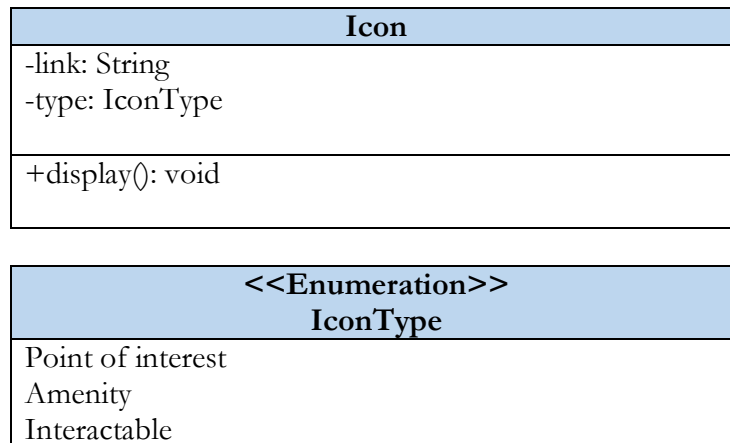
3.7.2.1.9 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Have a Filter screen.
3.2.5.7.1.1	The user may select “Scenic overlooks” or “Waterfalls” to be displayed on the Map Window.
3.2.5.7.1.2	The app displays the data in requirement 3.2.5.7.1.1 as icons overlaying the Map Window.
3.2.5.12	The app displays a “Search for Trails” button on the Map Window of the Map screen
3.2.5.12.1	Upon tapping the “Search for Trails” button, the app displays up to 20 trails in the area

3.7.3 Icon

Description: A visual indicator of a point of interest, amenity, or any kind of interactable map item. Generally associated with a map legend entity.

Class Diagram



3.7.3.1 Design Elements

3.7.3.1.1 link: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Store the link the icon will hold
Description	Stores the link that the icon will redirect to (if any) when tapped or clicked on in the application.

3.7.3.1.2 type: IconType

ITEM	DESCRIPTION
Type	Attribute
Purpose	Define what kind of Icon the icon is to define interaction.
Description	Define which type of icon the icon is to specify the interaction that the Icon will have.

3.7.3.1.3 display(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Displays the picture
Description	Updates the UI to display the icon with an optional link.
Parameters	None
Returns	Void
Functions Called	None

3.7.3.1.4 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.6.6.3.2	As the User navigates through the Drop-pin Map screen, the pin stays in the center of the screen
3.2.6.6.3.4.1	Upon tapping the picture of dropped pin location, the app navigates the User to the Map screen and display the pin on the map
3.2.6.8.5.1	If a User taps on the friend's icon on the Map screen, a pop-up Modal appears displaying the friend's name and the friend's timestamp
3.2.5.8	Upon Long-Pressing any icon on the Map Window of the Map screen, the app displays the 10-digit longitude and latitude of that icon

3.7.4 Map

Description: A visual representation of a real location often stylized or simplified for ease of use. This Map uses Google Maps API, and displays icons, a legend, trails, and friends to the user in real time. It features links to other views, and a search bar to allow quick navigation between distant locations. The map can also navigate the user to the trailhead once they locate a trail.

Class Diagram

Map
-mapLegend: MapLegend
+drawMap(): void

3.7.4.1 Design Elements

3.7.4.1.1 mapLegend: MapLegend

ITEM	DESCRIPTION
Type	Attribute
Purpose	Show meaningful information on the map.
Description	mapLegend contains the meaning of symbols used in the map. This is displayed with the map, showing icons and text that help the user to understand what the map is displaying.

3.7.4.1.2 drawMap(): void

ITEM	DESCRIPTION
Type	Function
Purpose	Draws the Google Maps map on the UI.
Description	Accesses Firebase to draw the map.
Parameters	None
Returns	None
Functions Called	firebaseAccess.queryFirestore(getGoogleMap())

3.7.4.1.3 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.1.1	Have a Map screen.
3.2.1.6	Upon tapping the “Map” icon on the Bottom Navigation Panel, the app navigates the User to the Map screen
3.2.1.7.1	The Settings screen has a map section.
3.2.5.1	The app has an interactive Map Window on the map screen.
3.2.5.2	The app shall display the user’s current altitude and the user’s profile icon as the User’s Current Location on the map screen.
3.2.5.2.1	The user’s device’s location determines if the altitude is measured in feet or meters.
3.2.5.2.2	The user may manually toggle the map view between feet and meters in the settings screen.
3.2.5.3	The map window of the map screen displays each trail as a colored line.
3.2.5.3.1	The trail colors will be determined by the difficulty of the trail.
3.2.5.3.1.1	Trails with a hard difficulty displays with an orange trail.
3.2.5.3.1.2	Trails with a moderate difficulty displays with a yellow trail.
3.2.5.3.1.3	Trails with an easy difficulty displays with a green trail line.
3.2.5.4	Upon long-pressing a trail, the app navigates the user to the trail description.
3.2.5.5	The Map Window displays the Trailhead icon at the Trailhead of the Trail.
3.2.5.6	The app displays a search bar on the Map screen.
3.2.5.6.1	The app centers the Map Window on the Map screen to the trailhead of the trail the User searched for and selected.
3.2.5.6.2.1	The app displays the Trails in the area the User searched for.
3.2.5.7	The app displays a filter icon in the Map screen header.
3.2.5.7.1	Upon tapping the filter icon in the Map screen, the user will be taken to the Filter screen.
3.2.5.10	The app displays a map type icon in the header of the Map screen
3.2.5.10.1	Upon tapping the map type icon on the Map screen, a Modal dropdown appears with two buttons: “Default” and “Satellite”
3.2.5.10.2	Upon tapping the Default button, the app alters the Map Window type of the Map screen to Default view
3.2.5.10.3	Upon tapping the Satellite button, the app alters the Map Window type of the Map screen to Satellite view
3.2.5.13	The Map Window of the Map screen is zoomable.

<i>SRS</i>	<i>Content</i>
3.2.5.13.1	The Map Window of the Map screen adjusts resolution based on the level of zoom.
3.2.5.13.2	The resolution increases on the Map Window of the Map screen when zooming in
3.2.5.13.3	The resolution decreases on the Map Window of the Map screen when zooming out
3.2.6.6.3.1	Upon tapping the Fab Button, the User is navigated to the Drop-Pin Map screen that is centered on the Trailhead of the selected Trail with a pin in the middle of the screen
3.2.6.6.3.2	As the User navigates through the Drop-pin Map screen, the pin stays in the center of the screen

3.7.5 MapLegend

Description: A controller that contains the meaning of symbols used in the map. This is displayed with the map, showing icons and text that help the user to understand what the map is displaying.

Class Diagram

MapLegend
-icon: Icon -symbolDefinition: String
+display(): void -readMapLegend(): int

3.7.5.1 Design Elements

3.7.5.1.1 icon: Icon

ITEM	DESCRIPTION
Type	Attribute
Purpose	Ease of use
Description	Visual representation of the symbols

3.7.5.1.2 symbolDefinition: String

ITEM	DESCRIPTION
Type	Attribute
Purpose	Provide user with symbol meanings
Description	Text that will describe what each symbol means

3.7.5.1.3 readMapLegend(): int

ITEM	DESCRIPTION
Type	Function
Purpose	Draws the Google Maps map on the UI.
Description	Accesses Firebase to draw the map.
Parameters	None
Returns	None
Functions Called	firebaseAccess.queryFirebase(getGoogleMap())

3.7.5.1.4 Design Concerns

<i>SRS</i>	<i>Content</i>
3.2.5.1	Have a Map Legend screen.
3.2.5.9	The app displays a legend icon in the header of the Map screen
3.2.5.9.1	Upon tapping the legend icon on the Map screen, the app navigates the User to the Map Legend screen
3.2.5.9.2	The Map Legend screen displays each icon followed by the corresponding meaning

4 Appendix

These are the requirements from the SRS that are not currently fulfilled by the SDD because they are part of the UserInterface and are out of scope.

<i>SRS #</i>	<i>DESCRIPTION</i>
3.2.1.11	The My Hikes screen shall have the following tabs: <ul style="list-style-type: none">• Current Hike tab• History tab
3.2.1.7	The Settings screen shall have the following sections: <ul style="list-style-type: none">• General section• Map section
3.2.1.8	The Browse screen shall have a Suggested Trails section
3.2.2.2	The app shall have a “Forgot my password” button on the Login screen
3.2.2.1.2.1	The Home screen icon is located on the app’s Bottom Navigation Panel
3.2.2.3	The app shall display a “Register for an account” button on the Login screen.
3.2.2.3.3	The Register screen shall display a “Register” button at the bottom of the screen
3.2.3.2	The app shall display a “Start” button on each Trail’s Trail Description screen
3.2.3.6.2	Each History Card shall not display the “Save” and “Stop” buttons.
3.2.5.6.2	The app shall center the Map Window on the Map screen to the search results the user searched for.
3.2.6.3.2	The post created by the User shall be displayed on the right $\frac{3}{4}$ of the thread
3.2.6.3.3	The post created by the User’s friends shall be displayed on the left $\frac{3}{4}$ of the thread
3.2.6.3.4	Each post shall display a vertical ellipses button
3.2.6.4	Each message thread will display a vertical ellipses button
3.2.5.6.2	The app shall center the Map Window on the Map screen to the search results the user searched for.
3.2.6.5.7.1	The “About Me” section shall display a vertical ellipses button in the top right corner of the section
3.2.6.6.3.4	In the read-only Community Forum screen, the User may tap on the picture of the dropped pin location in a post.
3.7.1.1	The app shall be available for download and installation on the Google Play Store or the App Store to Users with OS or Android platforms