CSCI 5448, Fall 2015

Hike Tracker

Part Two

Ryan Milvenan & Diana Southard

Hike Tracker

Table of Contents

Project Summary

This project is an Android application that tracks which of Colorado's famous "14ers" a user has hiked, how long each hike took, and overall average hiking length over all recorded hikes. The application will also present all mountain peaks locations along with brief summaries when any individual peak is selected which will include the mountain's actual height and which mountain range it is a part of.

Project Requirements

Business Requirements	
ID	**Requirement**
BR-001	Only Application Admin can change a mountain peak's name/elevation
BR-002	Application will accurately display all 53 14ers mountain peaks
User Requirements	
ID	**Requirement**
UR-001	User will see personal hiking summary on loading page
UR-002	User can click on "14ers" and see mapped location of all 14ers mountain peaks
UR-003	User can click on "History" and see list of saved hiked
UR-004	User can record a new hike.
UR-005	User can manually enter in a new hike.
UR-006	User can edit previously saved hike data.
UR-007	User can update displayed user name.
Functional Requirements	
ID	**Requirement**
FR-001	Application will display saved user name on landing page
FR-002	Application will be able to maintain timer throughout any-duration hike
FR-003	When saving new hike, "Peak Name" field will be populated by selected peak's name.

FR-004	When saving new hike, “Hike Date” field will be populated by current date.
FR-005	After correctly saving hike data, user will be redirected to application landing page.
FR-005	Application will correctly write data to private SQL database stored on phone’s internal memory
Non-Functional Requirements	
ID	**Requirement**
NFR-001	Application will load within 5 seconds.
NFR-002	During transition between activities, application will load new activity within 5 seconds
NFR-003	Application will have same behavior on different Android platforms
NFR-004	Application can be used by any level of user expertise

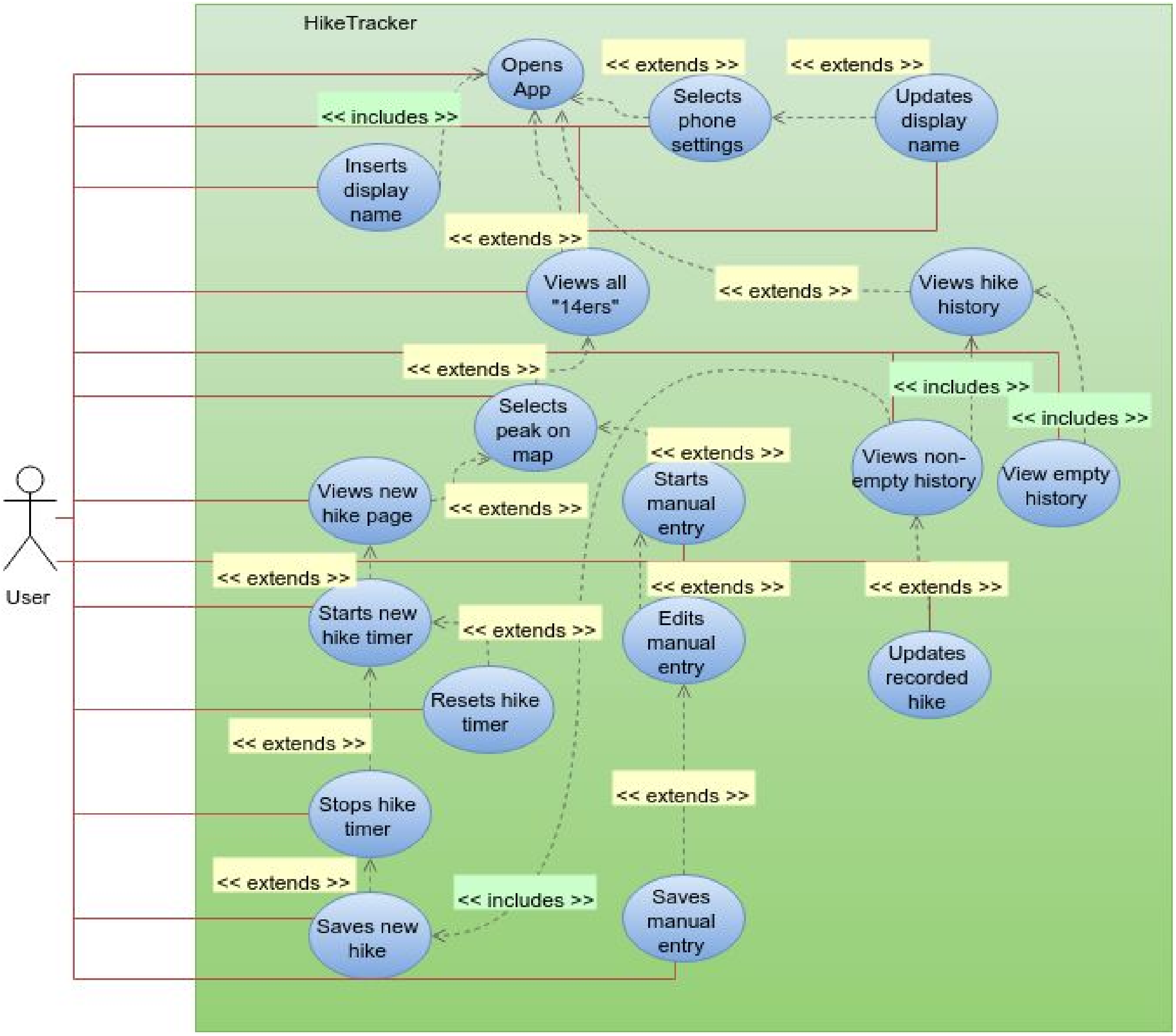
Data Storage

All data will be stored locally on the user’s smartphone. The stored data will consist of the user’s name, any recorded hikes (name of mountain peak, date of hike, length of hike, picture taken at the mountain summit), and current average hike time. Images will be saved to the phone’s local picture directory. User information will be saved using the SharedPreferences class provided by Android. Recorded hike information will be saved using a locally saved SQLite database which can only be accessed by the application. An SQLite database helper class will be created to access saved data at run-time as well as to store new records to the database.

Users and Tasks

The system will have a single user, the owner of the smartphone the application is currently running on.

Use Case Diagram



Use Cases

UC1: First-Time Opening Application

Actors: Smartphone user
Preconditions: Application has been successfully installed on smartphone.
Postconditions: Application is ready for new user name to be entered.
Frequency of Use: Once
Flow of Events:

1. User opens application from smartphone.
2. MainActivity initializes application landing page.
3. UserInfo attempts to load user's information.
4. UserInfo returns null user info.
5. MainActivity initializes NewUserActivity.
6. NewUserActivity displays prompt for user display name.

UC2: User Inserts Initial Display Name

Actors: Smartphone user
Preconditions: UC1 has been successfully completed
Postconditions: New user information has been saved
Frequency of Use: Once
Flow of Events:

1. User inserts desired display name in prompt.
2. User selects "Save" option in prompt.
3. UserInfo saves desired display name in SharedPreferences.
4. NewUserActivity closes.
5. MainActivity resumes.

UC3: User Opens Activity (Name Previously Saved)

Actors: Smartphone user
Preconditions: UC1 and UC2 have been successfully completed. Application is currently closed.
Postconditions: Application is successfully loaded
Frequency of Use: Each time application is opened (after UC2)
Flow of Events:

1. User opens application from smartphone.
2. MainActivity initializes application landing page.
3. UserInfo attempts to load user's information.
4. UserInfo successfully loads user's information.
5. MainActivity displays previously-stored display name, hiking summary, option to view hiking history, and option to view all "14ers"

UC4: User selects option to view all “14ers”

Actors: Smartphone user
Preconditions: UC3 has been successfully completed.
Postconditions: Map is displayed on screen with all desired peaks displayed as pins
Frequency of Use: Regularly
Flow of Events:

1. User selections option to view all “14ers.”
2. LocatorActivity loads map display with all “14ers” and user’s current location indicated as pins on map.
    a. Map displays any peak that has saved hike data with a pin indicating that peak has been hiked.
    b. Map displays any peak with no saved hike data with a pin indicating the peak has not been hiked.

UC5: User selects mountain peak pin on map

Actors: Smartphone user
Preconditions: UC4 has been successfully completed
Postconditions: Maps displays information box on screen
Frequency of Use: Regularly
Flow of Events:

1. User selects desired mountain peak to begin recording data by pressing that peak’s map pin.
2. LocatorActivity displays pop-up with name of mountain peak, peak’s elevation, option to start new hike, and option to manually input hike.

UC6: User view new hike page

Actors: Smartphone user
Preconditions: UC5 has been successfully completed
Postconditions: New hike page is displayed on screen
Frequency of Use: Regularly
Flow of Events:

1. User selects option to start new hike.
2. HikeActivity displays new hike page with new stopwatch counting length of this new hike and option to start/end hike.

UC7: User starts a new hike timer

Actors: Smartphone user
Preconditions: UC6 has been successfully completed
Postconditions: Hike timer is counting time
Frequency of Use: Regularly
Flow of Events:

1. User selects option to start the hike.
2. HikeActivity starts hike timer.

UC8: User resets hike timer:

Actors: Smartphone user
Preconditions: UC7 has been successfully completed, hike timer display is > 0
Postconditions: Hike timer has been reset to 0
Frequency of Use: Irregular
Flow of Events:

1. User selects option to reset hike timer.
2. HikeActivity resets hike timer to 0.
3. Hike timer continues to count hike time from 0.

UC9: User stops hike timer.

Actors: Smartphone user
Preconditions: UC7 has been successfully completed, hike timer display is > 0
Postconditions: New hike data has been created and is ready to be saved
Frequency of Use: Regularly
Flow of Events:

1. User selects option to end the hike.
2. HikeActivity creates new hike:
    a. HikeActivity stops stopwatch.
    b. HikeActivity records total length of hike.
3. HikeDialogFragmentActivity displays options to save or discard new hike record.

UC10: User saves new hike record

Actors: Smartphone user
Preconditions: UC9 has been successfully completed:
Postconditions: New hike record has been saved in database, application returns to landing page
Frequency of Use: Regular
Flow of Events:

1. User selects option to save new hike record.
2. Application saves new hike record.
    a. HikeDialogFragmentActivity adds new record to user's recorded hikes.
    b. UserInfo updates user's average hike time.
    c. UserInfo updates user's summit count.
    d. MountainList updates peak's hiked boolean to true.
3. Application returns to landing page with updated hiking summary

UC11: User discards new hike record

Actors: Smartphone user

Preconditions: UC9 has been successfully completed:
Postconditions: Application displays landing page, no new record has been saved
Frequency of Use: Rare
Flow of Events:

1. User selects option to cancel saving new hike record
2. Application returns to landing page with no change in hiking summary.

UC12: User selects option to manually enter in new hike

Actors: Smartphone user
Preconditions: UC5 has been successfully completed.
Postconditions: Application displays page for manually inputting hike data
Frequency of Use: Irregular
Flow of Events:

1. User selects option to manually input hike.
2. HikeDialogFragmentActivity displays page for inputting hike data.
    a. HikeDialogFragmentActivity displays spinners for selecting hike time length
    b. HikeDialogFragmentActivity displays date picker for selecting hike date.

UC13: User edits manual hike data

Actors: Smartphone user
Preconditions: UC12 has been successfully completed.
Postconditions: Manual data entry is entered and ready to be saved
Frequency of Use: Irregular
Flow of Events:

1. User manipulates spinner to input hike length.
2. User manipulates datepicker to input hike date.

UC14: User saves manual hike data.

Actors: Smartphone user
Preconditions: UC13 has been successfully completed
Postconditions: Manual entry has been saved in database, application returns to database
Frequency of Use: Irregular
Flow of Events:

1. User selects option to save manual hike record.
2. Application saves new hike record.
    a. HikeDialogFragmentActivity adds new record to user’s recorded hikes.
    b. UserInfo updates user’s average hike time.
    c. UserInfo updates user’s summit count.
    d. MountainList updates peak’s hiked boolean to true.
3. Application returns to landing page with updated hiking summary

UC15: User presses phone settings button

Actors: Smartphone user
Preconditions: UC3 has been successfully completed.
Postconditions: Settings menu is displayed
Frequency of Use: Irregular
Flow of Events:

1. User selects phone's settings button.
2. MainActivity displays option to update user display name.

UC16: User selects option to update display name

Actors: Smartphone user
Preconditions: UC15 has been successfully completed.
Postconditions: Application is ready to edit display name
Frequency of Use: Rare
Flow of Events:

1. User selects option to update user display name.
2. MainActivity displays pop-up prompting user to edit the current display name
    a. Pop-up displays editable text field.

UC17: User updates display name

Actors: Smartphone user
Preconditions: UC16 has been successfully completed.
Postconditions: New username is saved, application returns to landing page
Frequency of Use: Rare
Flow of Events:

1. User edits current user display name.
2. User saves edited display name.
3. UserInfo saves new user display name in the user's information.
4. MainActivity returns to landing page with new user display name shown

UC18: User selects hiking history

Actors: Smartphone user
Preconditions: UC3 has been successfully completed
Postconditions: Application is searching for any saved hike data
Frequency of Use: Regular
Flow of Events:

1. User presses on a history button
2. MainActivity initializes HistoryActivity.
3. HistoryActivity attempts to load saved hike data

UC19: User views empty hiking history

Actors: Smartphone user
Preconditions: UC18 has been successfully completed, no hike data has been saved to database
Postconditions: Application has returned to landing page
Frequency of Use: Rare
Flow of Events:

1. HikeDataSource finds no recorded hike data.
2. HistoryActivity displays message that no hikes have yet been recorded.
3. HistoryActivity displays option to return to MainActivity landing page.

UC20: User views non-empty hiking history

Actors: Smartphone user
Preconditions: Either UC10 (User saves new hike record) or UC14 (User saves manual hike data) has been successfully completed
Postconditions: Application displays list of saved hike data
Frequency of Use: Regular
Flow of Events:

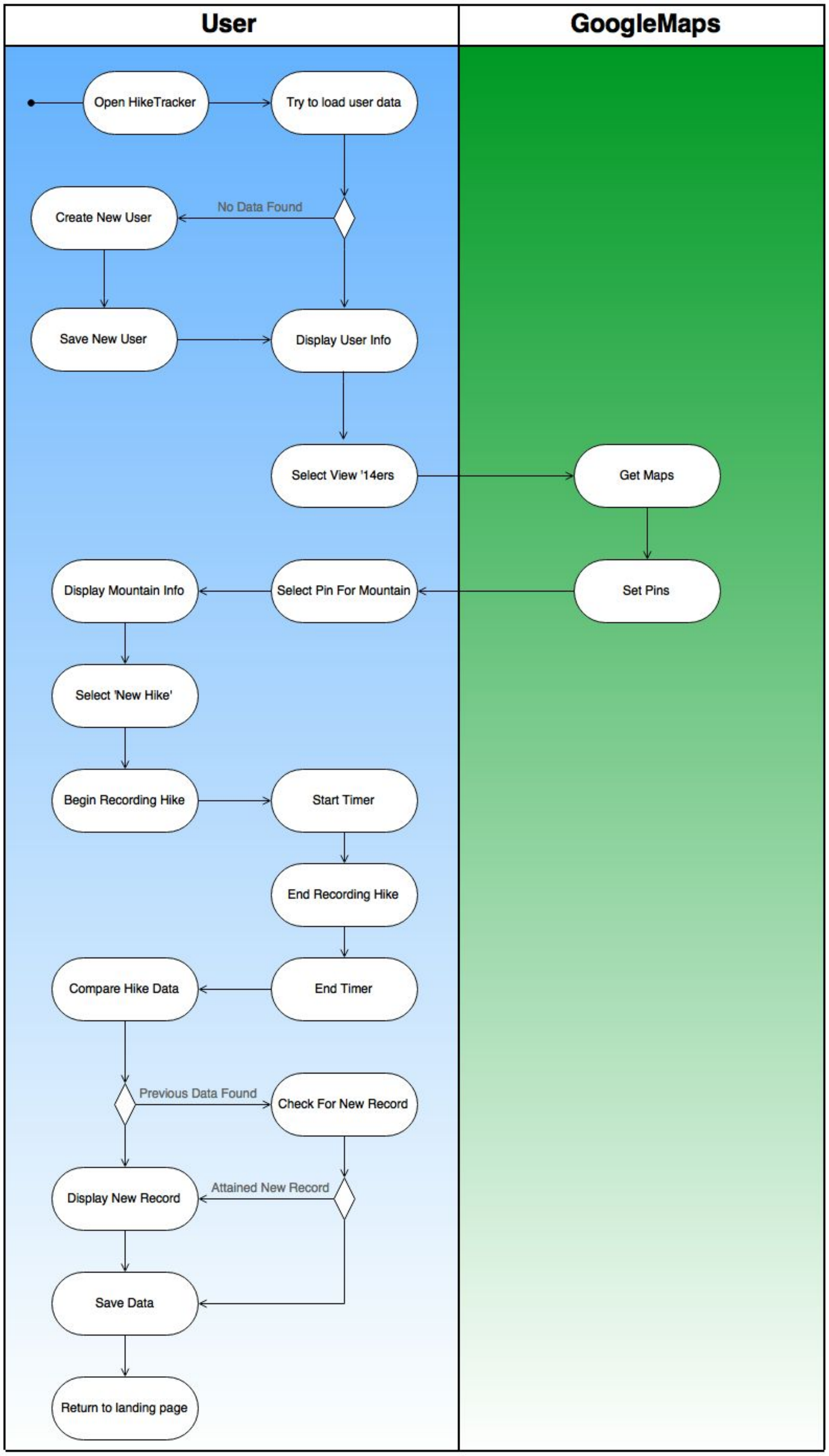
1. HikeDataSource finds recorded hike data.
2. HistoryActivity displays scrollable list of recorded hike data

UC21: User updates recorded hike

Actors: Smartphone user
Preconditions: UC20 has been successfully completed
Postconditions: Application loads manual entry page with previous hike data loaded
Frequency of Use: Rare
Flow of Events:

1. User presses on a recorded hike data.
2. HistoryActivity displays option to edit selected hike data.
3. User selects option to edit selected hike data
4. HikeDialogFragmentActivity displays page for inputting hike data.
    a. HikeDialogFragmentActivity displays spinners for selecting hike time length with previous hike time length displays
    b. HikeDialogFragmentActivity displays date picker for selecting hike date with previous hike date displayed

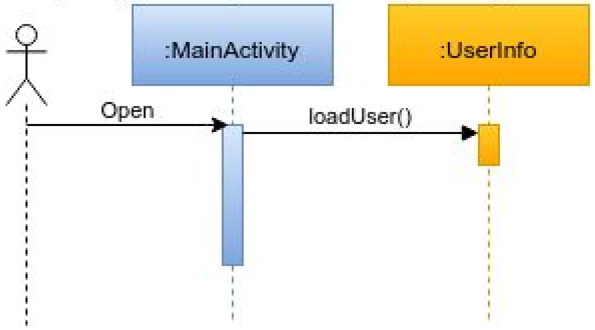
Activity Diagram



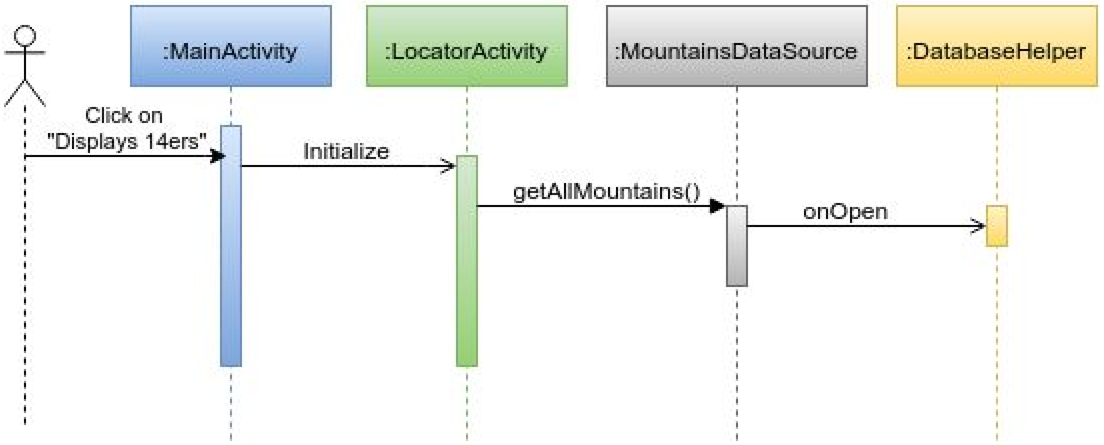
User Interaction

The user will have the following interactions with the project: opening the app, recording a new hike, manually entering a new hike, updating previously saved hiking data, updating their displayed user name, and closing the app. The main interaction will be the user recording a new hike. See the sequence diagram below to show how the application classes will interact during this exchange.

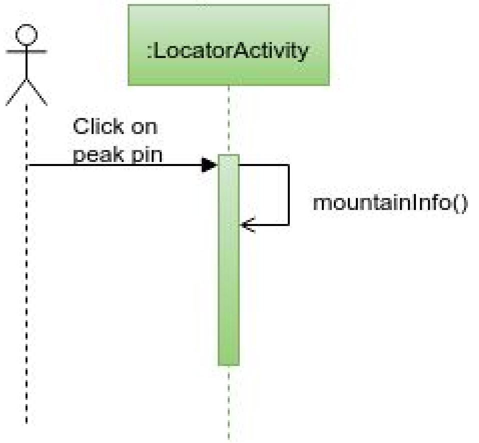
Open Application (Name Previously Saved)



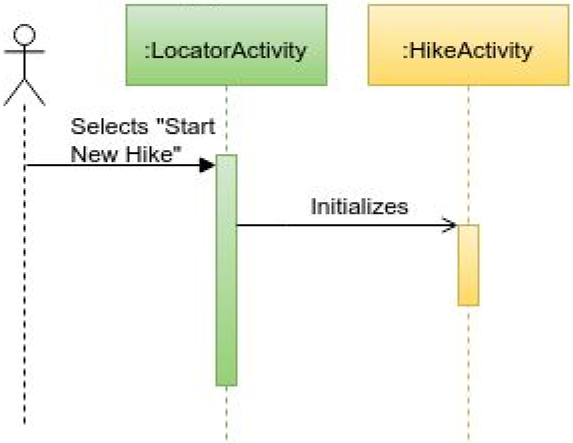
Display all "14ers"



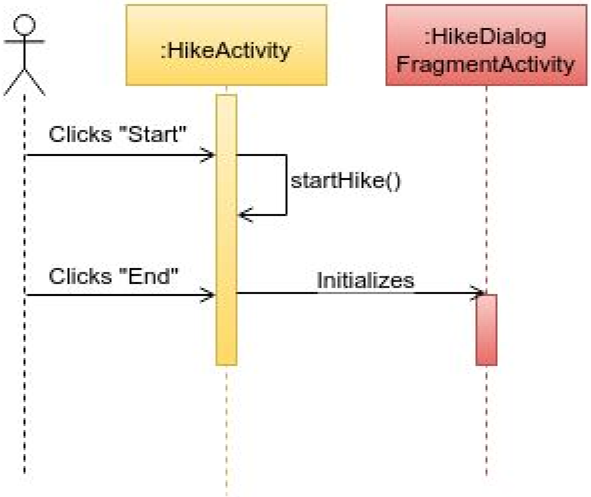
Selects Pin on Map



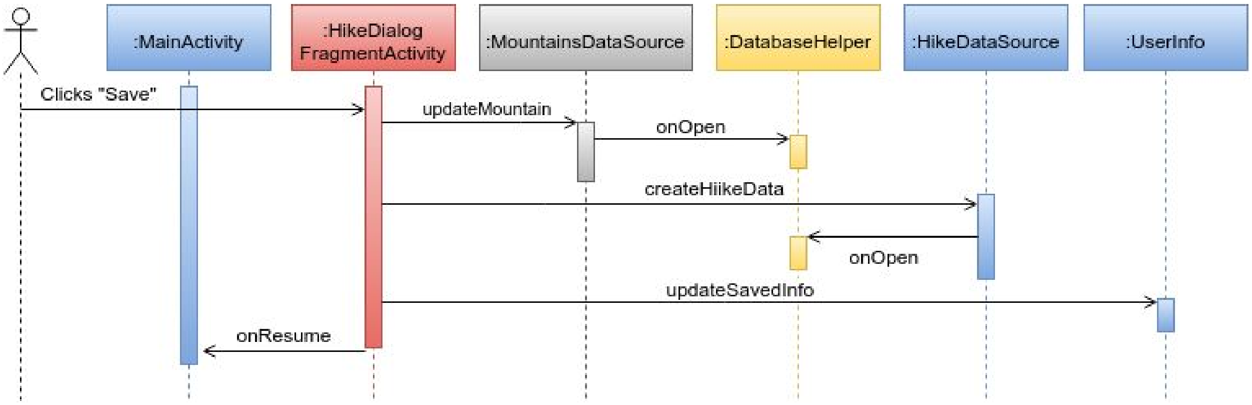
View new hike page



Start/End Hike

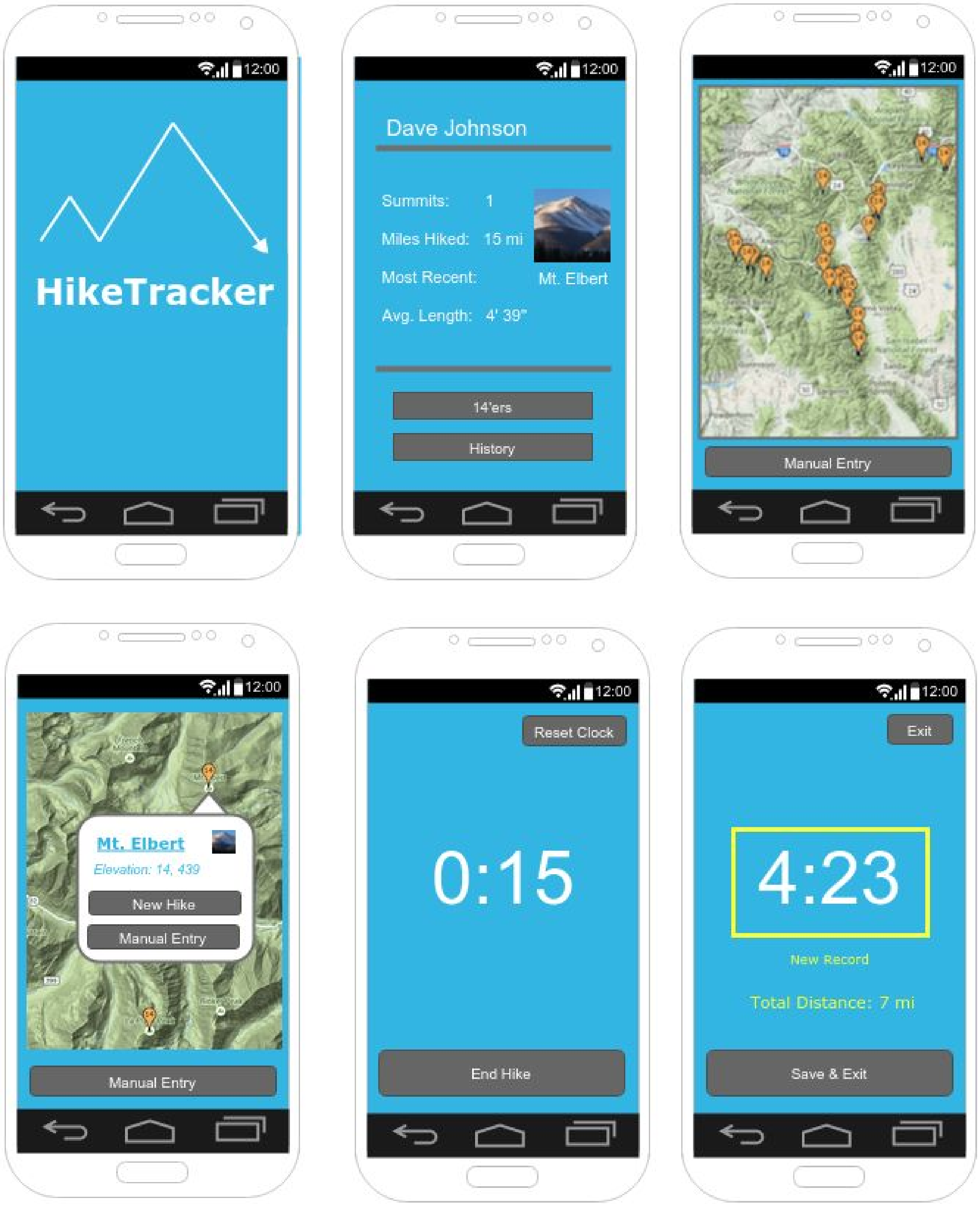


Save new hike record



UI Mockups

Below are some mock-ups displaying the screens for the main scenario of the user starting to record a new hike. The screens are displays in sequential order, from left to right.



Dave Johnson

Summits: 1

Miles Hiked: 15 mi

Most Recent: Mt. Elbert

Avg. Length: 4' 39"

Mt. Elbert

Elevation: 14, 439

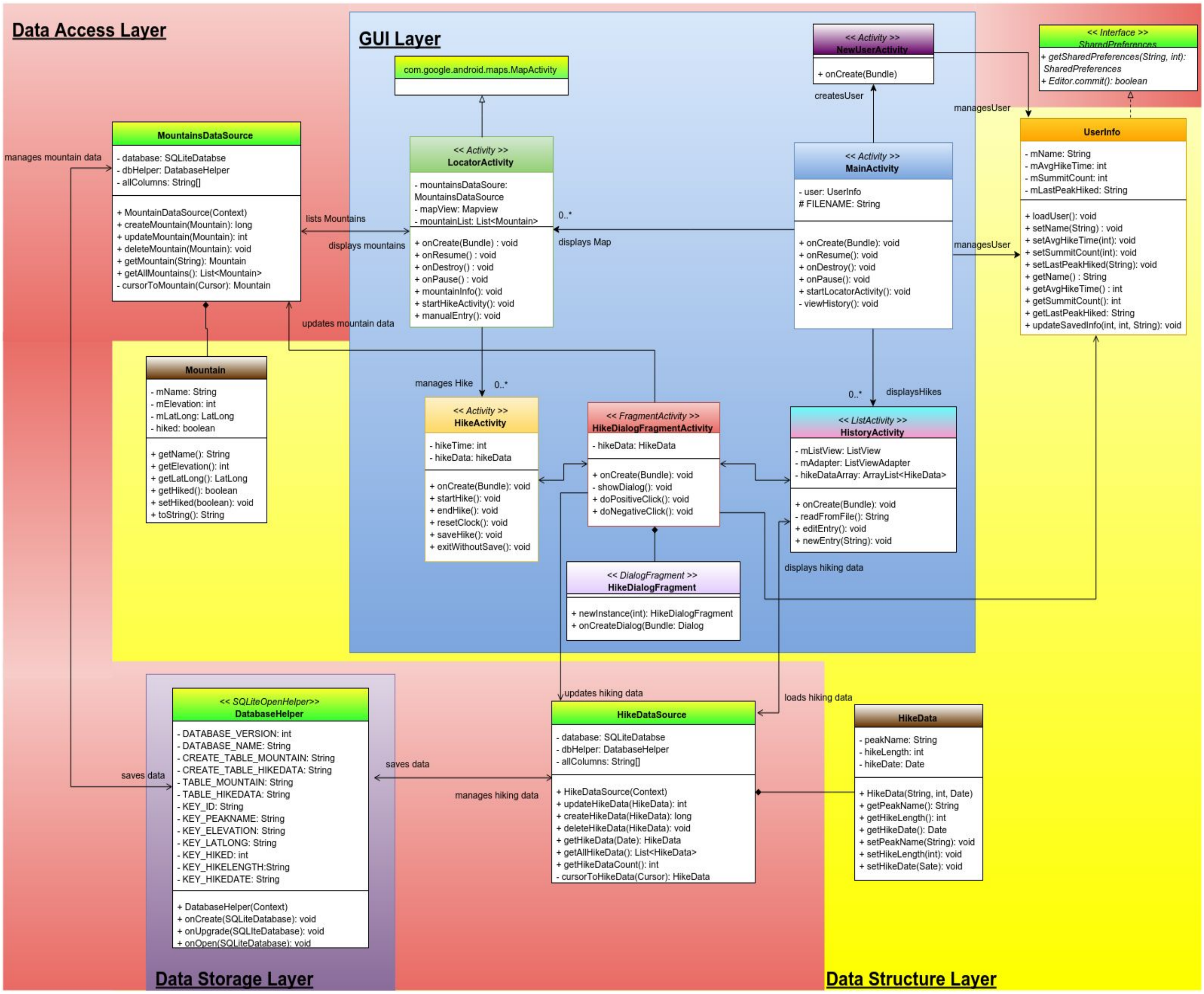
0:15

4:23

New Record

Total Distance: 7 mi

Class Diagram


Data Access Layer
GUI Layer
Data Storage Layer
Data Structure Layer
com.google.android.maps.MapActivity
MountainsDataSource
- database: SQLiteDatabse
- dbHelper: DatabaseHelper
- allColumns: String[]
+ MountainDataSource(Context)
+ createMountain(Mountain): long
+ updateMountain(Mountain): int
+ deleteMountain(Mountain): void
+ getMountain(String): Mountain
+ getAllMountains(): List<Mountain>
- cursorToMountain(Cursor): Mountain
manages mountain data
lists Mountains
displays mountains
updates mountain data
<< Activity >>
LocatorActivity
- mountainsDataSoure: MountainsDataSource
- mapView: Mapview
- mountainList: List<Mountain>
+ onCreate(Bundle) : void
+ onResume() : void
+ onDestroy() : void
+ onPause() : void
+ mountainInfo(): void
+ startHikeActivity(): void
+ manualEntry(): void
0..*
displays Map
<< Activity >>
NewUserActivity
+ onCreate(Bundle)
createsUser
managesUser
<< Interface >>
SharedPreferences
+ getSharedPreferences(String, int): SharedPreferences
+ Editor.commit(): boolean
<< Activity >>
MainActivity
- user: UserInfo
# FILENAME: String
+ onCreate(Bundle): void
+ onResume(): void
+ onDestroy(): void
+ onPause(): void
+ startLocatorActivity(): void
- viewHistory(): void
managesUser
UserInfo
- mName: String
- mAvgHikeTime: int
- mSummitCount: int
- mLastPeakHiked: String
+ loadUser(): void
+ setName(String) : void
+ setAvgHikeTime(int): void
+ setSummitCount(int): void
+ setLastPeakHiked(String): void
+ getName() : String
+ getAvgHikeTime() : int
+ getSummitCount(): int
+ getLastPeakHiked: String
+ updateSavedInfo(int, int, String): void
Mountain
- mName: String
- mElevation: int
- mLatLong: LatLong
- hiked: boolean
+ getName(): String
+ getElevation(): int
+ getLatLong(): LatLong
+ getHiked(): boolean
+ setHiked(boolean): void
+ toString(): String
manages Hike
0..*
<< Activity >>
HikeActivity
- hikeTime: int
- hikeData: hikeData
+ onCreate(Bundle): void
+ startHike(): void
+ endHike(): void
+ resetClock(): void
+ saveHike(): void
+ exitWithoutSave(): void
<< FragmentActivity >>
HikeDialogFragmentActivity
- hikeData: HikeData
+ onCreate(Bundle): void
- showDialog(): void
+ doPositiveClick(): void
+ doNegativeClick(): void
0..*
displaysHikes
<< ListActivity >>
HistoryActivity
- mListView: ListView
- mAdapter: ListViewAdapter
- hikeDataArray: ArrayList<HikeData>
+ onCreate(Bundle): void
- readFromFile(): String
+ editEntry(): void
+ newEntry(String): void
displays hiking data
<< DialogFragment >>
HikeDialogFragment
+ newInstance(int): HikeDialogFragment
+ onCreateDialog(Bundle: Dialog
<< SQLiteOpenHelper>>
DatabaseHelper
- DATABASE_VERSION: int
- DATABASE_NAME: String
- CREATE_TABLE_MOUNTAIN: String
- CREATE_TABLE_HIKEDATA: String
- TABLE_MOUNTAIN: String
- TABLE_HIKEDATA: String
- KEY_ID: String
- KEY_PEAKNAME: String
- KEY_ELEVATION: String
- KEY_LATLONG: String
- KEY_HIKED: int
- KEY_HIKELENGTH:String
- KEY_HIKEDATE: String
+ DatabaseHelper(Context)
+ onCreate(SQLiteDatabase): void
+ onUpgrade(SQLIteDatabase): void
+ onOpen(SQLiteDatabase): void
saves data
saves data
manages hiking data
updates hiking data
loads hiking data
HikeDataSource
- database: SQLiteDatabse
- dbHelper: DatabaseHelper
- allColumns: String[]
+ HikeDataSource(Context)
+ updateHikeData(HikeData): int
+ createHikeData(HikeData): long
+ deleteHikeData(HikeData): void
+ getHikeData(Date): HikeData
+ getAllHikeData(): List<HikeData>
+ getHikeDataCount(): int
- cursorToHikeData(Cursor): HikeData
HikeData
- peakName: String
- hikeLength: int
- hikeDate: Date
+ HikeData(String, int, Date)
+ getPeakName(): String
+ getHikeLength(): int
+ getHikeDate(): Date
+ setPeakName(String): void
+ setHikeLength(int): void
+ setHikeDate(Sate): void