Discovr - Test Plan Document

Introduction

Discovr is an android app that integrates mapping and directional capabilities with an events and obstacle database. This app provides UBC students and its residents with a graphically interactive map containing relevant information that cannot be easily found or accessed from a search and events held within UBC campus.

Verification Strategy

To obtain feedback on the product we conduct user testing sessions with fellow students and friends; during these sessions they will be presented with 3 tests:

1. Ease-of-use/Learning Curve - Without given context as to what the app does, we will give the users a short time to interact with it. After the given timeframe has passed we will explain what the app is suppose to do in a broad sense. We will give them a task such as "Find an event" without explaining the steps on how to achieve it in order see how intuitive/non-intuitive our UX.
2. A/B test - Given two mockups of a UI we will ask the user accomplish a task with it and observe which one they can accomplish with ease without any help; the point of this test is to help choose which UI prototype is a suitable candidate to be further implemented.
3. Open Feedback - We then ask the user if the app provides the functionality that they expected based on our description.

Non-Functional Testing and Results

Non-Functional Requirements:

1. The app should not drain a huge portion of the user's battery and battery usage should be comparable to other similar apps like Google Maps
2. The app should try to reduce network usage when loading information from remote servers, especially when using mobile data
3. Application should notify users to be aware of surrounding conditions and give details to the UBC safewalk program when using map features past sunset hours
4. Secure the API Key for for both Mapbox and Translink to prevent external users from using up all of our API transactions
5. Secure remote database connection string to prevent malicious access to our data
6. The app must be reliable and robust such that the user can depend on it on their daily workflow
7. The app must be simple and straightforward such that information can easily be found and doesn't have a high requirement from the user

Test #	Requirement Purpose	Action / Input	Expected Result	Actual Result	P/F
1	Performance	Keep application running in foreground for twenty minutes	Battery should not decrease by a significant amount	App does not significantly affect the battery usage compared to system use.	Pass
2	Performance	Use application to navigate for twenty minutes and scroll through 100 events	Phone should use no more than 30mb of data	Phone does not use a	Pass
3	User Safety	Set time to 10:00 PM and open application	User should be notified to be aware of walking alone and given an option to call safewalk	User should be notified to be aware of walking alone and given an option to call safewalk	Pass
4	Security	Set a limit for API transactions for each user without reducing the software quality	Users who demands more API transactions will have their requests be delayed	API transaction is of no issue for now so there are no limits to the amount of transactions done	Half
5	Security	Try to access the database from a non-user terminal	The non-user terminal should fail to access data in our remote db.	The database cannot be accessed unless going through our site or through approved http requests	Pass
6	Software Quality	Present 25 students with beta release of application	90% of testers should be able to use and navigate through the application with no context and help	Application can be used by most students without an issue	Pass

Functional Testing Strategy

To test the software we will use different strategies at different phases of development:

Unit Testing: Most of the issues we created are considered to be individual 'units' which combines to implement a functionality. Each unit will be tested before being merged into the master branch. Units will also undergo fault-based testing, specifically targeted at likely bugs.

Integration Testing: Integration testing is done each time a development branch is merged into the master branch. It involves examining the functionality more than one unit working together. Performing integration testing can help us detect bugs and errors when data are passed between units within the system.

Systems Test: We perform a systems test each time a core feature is completed. We check whether the feature meets the requirement specification for this feature while ensuring the non-functional requirements as a whole are met as well.

Github has an project issue section which we use as a To-do list. This section contains entries on what we need to work on to implement the app's core functionalities as well as bugs that we encounter during this process. Each entry in this section can be tagged correspondingly, for example, with 'bug' or 'extension'. Issues can be closed manually or linked alongside a pull request which fixes or implements the issue. Once the pull request is merged alongside the master branch, the issue is closed automatically.

Adequacy Criterion

1. Make sure that at least one test exists for each unit (issue)
   We need to make sure each issue is solved properly before we add complexity to its environment (use them in more complex contexts).
2. Make sure that at least one test exists for each requirement
   We need to make sure that our product meet each requirement as expected.
3. Make sure that at least one test exists for each use case
   We need to ensure that our app is practical for in each use case before we consider a more complicated case.
4. Make sure that at least one test exists for each activity class
   We need to make sure each "activity" works as expected before we integrate them as a whole

Test Cases:

Functionality	Issue ID	Action / Input	Expected Result	Actual Result	P/F
UI	5	User taps on a button to navigate different sections of the app	User is able to access the different views (Map, Events, etc)	User can use the navigation drawer to access the views	Pass
UI	22	User switches from the map to another view and switches back to the map	The map remembers its location	The map fragment is hidden, not destroyed, and then shown when returning so no data is lost	Pass
UI	14, 24	User selects the search bar when on map view and types a location	User receives search suggestion and submitting query results in the location being found on the map	User is able to type in a location and receives results with location on the map. Though sometimes the results are a miss.	Pass
UI	N/A	User selects an event on their events page	Details of the events are brought up alongside its location on the map	A fragment appears and is overlaid on bottom of their map	Pass
Building	63	User selects a building on the map	Details of the selected building should be brought up	A fragment appears and is overlaid on bottom of the map	Pass
Course	27	User uploads an iCal of their class schedule.	iCal4j loads iCal info to our program and all the entries on are stored as "Course" type	All the courses can be seen by the user under their courses list	Pass
Course	N/A	User selects a course under their course list	User is brought to the map with course information provided	User is brought to the map with course information provided	
Course	118	N/A	An alert shows up if the user is currently on the "List All Courses" page and there exists some course starting in 10 minutes	The notification shows up with the coming course title	Pass
Course/Event	75,125	The map location should be moved to the building when user selects specific event/course	The map is moved to the building of the selected event/course	The map move to the location where the event/course is holding	Pass

Event	N/A	User subscribes to an event from the event list page	User should see the event if they navigate to their list of subscribed event	User sees the event when navigated to from their list of subscribed events	Pass
Event	N/A	User unsubscribes from an event	User should not see the event on their subscribed list nor receive further notification about the event	User does not see the unsubscribed event from their subscribed page	Pass
Event, RemoteDB	N/A	User creates an event and submits it to the app	The event appears on the app after being reviewed by a moderator and the creator receives a special key to modify the event in the future	The event appears on the app after being reviewed by a moderator and the creator receives a special key to modify the event in the future	Pass
Event, RemoteDB	N/A	Event creator enters the key and modifies their event	Event is updated and all subscribed user receives a notification of the change	Event is updated and the user can view the updated events in the All Events list. User does not recieve notification.	Half
MB	4, 31	User selects the Floating Action Button	The map focuses on the current location of the user	The map focuses on user's location with corresponding pin	Pass
MB	N/A	User selects 'Direction' on an event or building	A waypoint navigation system is shown on the map directing user to their target	Selecting an event or building will automatically show directions to the user	Pass
MB	N/A	User opens a map and navigates to an area with construction nearby	The construction area is highlighted on the map	The constructed area is highlighted on the map	Pass
MB	N/A	User selects Upcoming Events	The system displays a map with markers where upcoming events are held	The system displays a map with markers where upcoming events are held	Pass
Local DB	6	User adds event to list of subscribed events	User is able to access list of subscribed events from localDB without request from remote DB (without data connection)	User is able to access list of subscribed events from localDB without request from remote DB (without data connection)	Pass
Transit	N/A	User selects Show Buses	The map displays markers where bus stops are	The map displays markers where bus stops are	Pass
Transit	N/A	User selects a bus marker	A fragment appears showing the time for upcoming buses	A fragment appears showing the time for upcoming buses	Pass