**CS673 Software Engineering** 

**Team 1 - HoopFinder**

**Project Proposal and Planning**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Mike Zhong | Design, Implementation Leader  Backup Team Lead |  |  |
| Sriram Doss | Requirements Leader |  |  |
| Jamie Smart | QA Leader |  |  |
| Douyao Zhang | Security Leader |  |  |
| Saloni Rawat | Configuration Leader,  Team Lead |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| **00** | **Mike Zhong** | **15SEP2019** | **Initial revision** |
| **01** | **Saloni Rawat** | **26 Sep 2019** | **Added details under Configuration Management Plan** |
| **02** | **Sriram Ramdoss** | **28 Sep 2019** | **Added proposed High level Requirements** |
| **03** | **Jamie Smart** | **01 Oct 2019** | **Added details under Quality Assurance Plan** |
| **04** | **Saloni Rawat** | **11 Oct 2019** | **Added project logo, Monitoring and Controlling Mechanism, schedule and deadlines.** |

[Overview](#_2bn6wsx)

[Related Work](#_qsh70q)

[Detailed Description](#_3as4poj)

[Management Plan](#_tyjcwt)

[Process Model](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.27177f40uci)

[Risk Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.a4oqwntk3mw)

[Monitoring and Controlling Mechanism](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.ywdoc2clc9yt)

[Schedule and deadline](#_1pxezwc)

[Quality Assurance Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.72e1f4uawy2r)

[Metrics](#_49x2ik5)

[Standard](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.vc72k6dweldv)

[Inspection/Review Process](#_1ksv4uv)

[Testing](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.r5d5mhtlf0kq)

[Defect Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.54a4wuncjg1c)

[Process improvement process](#_2p2csry)

[Configuration Management Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.hw41vg4ykxen)

[Configuration items and tools](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.bwlb4d4vdox2)

[code commit guidelines](#_2xcytpi)

[References](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.8mva2050iy7t)

[Glossary](#_3whwml4)

# Overview

The purpose of the “HoopFinder” app is to facilitate organization and involvement in pick-up sports amongst strangers, specifically basketball in this case but the app can be extended for any sport and/or exercise.

There are many basketball courts in a given city and most of the time, they are not being used. On average, a casual basketball player will spend 1-3 hours at a court, the likelihood a group of players 6+ players all overlap is very low unless players show up in groups, often of 2-3. This application will facilitate the organization of pick-up basketball games at local courts without requiring you to know the other players involved personally.

The application maps out basketball courts (or baseball, soccer, etc...) and uses user location services to identify when a user is "near" a court. This will trigger an alert to be sent to all other users (subscribed? friends? proximity?) that a fellow baller is available.

Users will register their devices with the application and whenever another player approaches a basketball court near them, they will receive a notification. Users can subscribe to courts and/or other players to get more relevant notifications.

# Related Work

This application will function similar to the Four Square app, which allows users who approach and enter establishments to “Check-in” with said establishment, provided the establishment has registered with the company. The company likely keeps a database of all registered “hosts” and using user location services, allows users to check-in to a host location upon arrival.

The major difference will be that this application will not require hosts to register, rather, a “host” in this case will be a basketball court (or any other public park). The locations of basketball courts will have to be scraped up-front and a database of geographic coordinates assembled.

# Proposed High level Requirements

* 1. **Functional Requirements**

i . Essential Features - the core features that you definitely need to finish

1. Use google play services API to return user location information either as a response to a button click, or on a set interval
2. A user should be able to create an account and login
3. A user can be able to subscribe court(host)
4. A user shall be able to get pop-up notification when another user is close to the court

ii. Desirable Features - the nice features that you really want to have too

1. A user shall be able to subscribe/follow other users
2. A user shall be able to alert other users that they are planning to join them at a court
3. A user can add new courts to database

iii. Optional Features - additional cool features that you want to have if there is time

1. A user should have option for social login
2. A user shall be able to get pop-up notification when new court is added

iv. Existing Features - Not applied to a brand new project

1. Foursquare : By using user location service, they will show nearby restaurants, nightlife spots, shops to the users
2. Find My Friends : By using GPS in the iOS device,User can be tracked.This application is to track children, family, and friends
   1. **Nonfunctional Requirements**
      1. Performance

For certain time interval (5 minutes) user location is checked and updated

* + 1. Use-ability

User can be able to understand the flow of HoopFinder easily without any guideline

* + 1. Reliability

User should give permission to access location and SMS notification on HoopFinder

* + 1. Security

Firebase authentication token is used for comparison and security

Purpose

* + 1. Storage
    - F
  1. **Implemented Features**

# Management Plan

# (For more details, please refer to SPMP document for encounter example)

## Process Model

As a process model, the team has decided on an iterative approach - Agile. Given the short amount of time for the project, it feels best suited for our needs.

According to the agile model, a select features will be chosen each iteration to be completed entirely (including testing) ready to be integrated with the application. At the end of each iteration, there will be a new version released with some new features. Given the agile model, we will quickly able to adapt to any change to the requirements/scope of the project giving us the flexible to provide a better quality product.

## Objectives and Priorities

(Project Goals can include but not limited to complete all proposed (essential) features, deploy the software successfully, the software has no known bugs, maintain high quality, etc )

The project goals for the team are:

1. Deploy software successfully
2. The software has no known bugs
3. Well documented project and code
4. Make navigation as simple as possible

## Risk Management (need update constantly)

.

The risk management document for the projects can be found on the link below. The risks have been identified and categorized under the following categories:

1. Personal

2. Communication

3. Requirement

4. Development

5. Management

6. Testing

7. Integration

8. Deployment

A brief title and description is provided for each risk item. Estimated likelihood of occurrence is decided based on the following categories:

H - High - >75% of occurance

M- Medium - < 75% & >40% of occurance

L - Low - < 40% of occurance

The amount of rework required is decided based on the following parameters:

H - High - > 8hrs of rework

M - Medium - <8hrs & >2hrs

L - Low - <2hrs

The legend for each column is provided under the “Legend” tab of the spreadsheet. The entire document can be found at the following link:

[https://docs.google.com/spreadsheets/d/1BsizjOC6WBk\_iK5iA5BvmczAlPb2INCN/edit#gid=436921](https://docs.google.com/spreadsheets/d/1BsizjOC6WBk_iK5iA5BvmczAlPb2INCN/edit#gid=436921088)

[088](https://docs.google.com/spreadsheets/d/1BsizjOC6WBk_iK5iA5BvmczAlPb2INCN/edit#gid=436921088)

## Monitoring and Controlling Mechanism

1. Weekly meetings and status reports to discuss the progress made by the group.
2. Quick status check after the class
3. Update the weekly progress report regularly.
   1. Team members will meet intermittently to discuss progress of individual tasks.
4. GIT branches are merged in the presence of the whole group.
   1. Incase, urgent merges need to be made to the Master, the merge should be approved by another team member to avoid conflicts.

**Communication**:

* All information to be communicated using Slack. Team members to ensure that they have the notifications enabled as well downloaded the mobile application, to be available for urgent issues.
* Phone calls/ texts to be used to extremely urgent case where a team member is not reachable via Slack.

**Pivotal Tracker:**

* PivotalTracker to be used to track progress of the individuals as well as team on a weekly basis.
* Only items being worked on in the current iteration to be updated in the current backlog. Remaining all use cases to be in Icebox.
* To store all the stories on PivotalTracker with the following information:

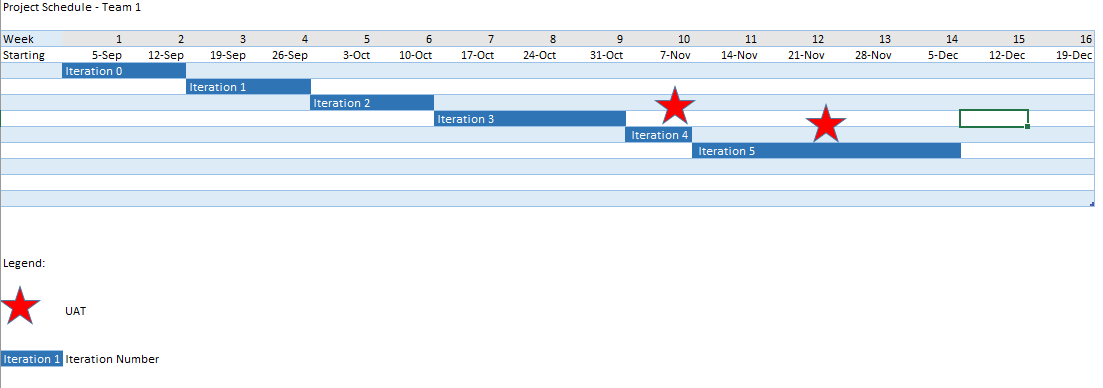
.

|  |  |  |
| --- | --- | --- |
| **Title** | **Data** | **Frequency** |
| Status | Started -- work on story has started.  Finished – Development completed.  Delivered – Ready for testing.  Rejected – Rework post testing required.  Accepted – Passed all testing. Ready to deploy | Weekly  OR  When there is a change in state. |
| Reviewers | Individual responsible for testing (doesn’t necessarily have to be the developer) Same as SIT.  10/2: To be discussed closer to testing  10/16: Before the end of the iteration, name of the tester. Could be the developer or another team member. | Beginning of the Iteration in which the story will be tested. |
| Story | - Feature  - Bug  - Chore  - Release | When the story is created. |
| Points | - 0  - 1  - 2  - 3 | When the story is created. |
| Requester | The developer working on the user story | Prior to beginning work on Story |
| Owner | The developer working on the user story | Prior to beginning work on Story |

## Schedule and deadlines (need update constantly)

The project schedule has been divided based on the schedule provided. There will 5 iterations in total. The graphical respresentation is available on - <https://docs.google.com/spreadsheets/d/1foJsS1XkU5BejmGL7SC7rto-LRIjIu9_/edit#gid=1077425432>.

A screenshot is attached below for reference.



Each iteration has been broken down into expected tasks to be completed within the iteration as week as any other important dates. You can refer to the ‘Iteration breakdown” in the project management.xls available on

<https://docs.google.com/spreadsheets/d/1foJsS1XkU5BejmGL7SC7rto-LRIjIu9_/edit#gid=1077425432>. The table is also included below for reference.

# Quality Assurance Plan

# (For more details, please refer to SQAP document for encounter example)

## Metrics

* + 1. Definition (e.g. define what metrics will be used, how to keep track of metrics, and how to analyze the metrics for process improvement. Two types of metrics should be included: product metrics and process metrics. Particularly include product complexity (LOC, # of files, # of classes, # of methods etc.) cost (in terms of man hours), defect and defect fix rate etc.user story points,
* track number of coding hours per iteration
* track number of files
* ≤ 5 defects / iteration
* add 500 new lines of code per iteration
* 1+ new user story per iteration

metrics will be analyzed to identify bottlenecks in the process

* + 1. Results (to be completed at the end of each iteration),

Results

* Need more testing to find more defects
* Need to implement user stories more quickly
  1. Standard  
     (e.g. documentation standard, coding standards etc. )

Camel case for method names

Capitalize first letter of each word in a class name

Generate java doc comments as useful

Class files should contain a comment description in the header and other comments about the content of the file as useful/needed

## Inspection/Review Process (e.g. describe what are subject to review, when to conduct review, who does the reviews and how ?)

Updated (11/16)

* All java class files to be peer reviewed before merging with dev branch
* Work will be done on individual branches, when work is ready for merge, the author will initiate a pull request with the code’s intended purpose and any additional things the reviewer should be aware of/check for
* Reviews will be assigned to one person as needed based on team availability
* Any needed changes will be corrected by original author before pull request is accepted by reviewer and writer merges change into dev branch

## Testing: (e.g. who, when and what type of testing to be performed? How to keep track of testing results?)

(Updated 11/16)

Unit Tests:

* 10+ unit tests per class

Integration Tests:

* Perform 2 tests for each method in a class that interacts with a method or object in another class

Functional Tests:

* These will be performed ad hoc during development to test functionalities currently under development

Non-functional Tests:

* Usability
  + measure number of clicks to get to various activities
  + test GUI display on different devices
  + gather non-developer ratings on ease of app use
* Security
  + Test password acceptance and storage
  + Test for possibility of injection attacks

Regression Tests:

* When one class or activity is updated, all other code in other classes that depends on the changed code should be re-tested before merging with dev

UAT:

* Before release, the app will be field-tested by the team and checked against the requirements
* Someone not on the programming team will be given a list of user stories and also use the app to confirm that the user stories can all be performed

*A separate document about testing results should be linked here.*

10/16SR : document can be found at [*https://docs.google.com/spreadsheets/d/1JHxTLIY\_rFV8pKoF4v0wxqOslVPpMcLoxlkhfdxNi0A/edit#gid=0*](https://docs.google.com/spreadsheets/d/1JHxTLIY_rFV8pKoF4v0wxqOslVPpMcLoxlkhfdxNi0A/edit#gid=0)

## Defect Management (e.g. describe the criteria of defect, also in terms of severity, extend, priority, etc. The tool used to management defect, actions or personnel for defect management)

A defect shall be considered to be any unintended behavior of any part of the app that prevents it from functioning as intended.

* Severe defects - a requirement is not met
* Moderate defects -requirements are met, but operation or maintenance is affected
* Minor defects - defect doesn’t impact operation or maintenance

Defects shall be reported as part of the peer review process and also when the

process of pushing the dev branch to master begins

The current defect tracker can be found on <https://docs.google.com/spreadsheets/d/1mzWnhIPcWQ5inY26V283iGa4PtKcEey0SCy56Air8cE/edit#gid=0>

Defects are also logged and tracked in the GitHub repository and can be viewed on the following link - <https://github.com/bumetcs673f19/cs673-project-team-1/issues>

# Configuration Management Plan

(For more details, please refer to SCMP document for encounter example)

## Configuration items and tools

**Code Versioning:**

* All versions of the code to be stored on Github.
* Master to contain the latest working copy of the code.
* New branches for each feature to be added.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Branch Name | Description | Created/Maintained by | Status | Comments |
| Documentation | To keep all updated copies of the documentation | Saloni Rawat | Active | 10/2SR: Merge with master after end of every iteration |
| Feature-Login | Dev copy of the app with the login feature | Saloni Rawat | Active | 10/2: Merge with master once the completed testing |
| Lab1 | Archive Lab1 folder |  | Archived | 10/2: Archived post iteration 0 |
| dev | Buffer branch to merge before Master | Mike Zhong | Active | 10/17: Up to date with Iteration 2 code. |
| master |  |  |  |  |
| project-skeleton |  |  |  |  |
| push-notify |  |  |  |  |
| user-location |  |  |  |  |

## Change management and branch management

* For each feature, a new branch will be created to allow multiple team members to work on the different user stories simultaneously and avoid conflicts.
* Once development is completed, the branch will be merged to Master and should be pulled by all the team members to avoid conflict.
  + New branch ‘Dev’ created to be a sandbox version of the merged code from the current version.
  + Once all the code has been validated, Dev will be merged to Master to have final version without any conflicts.
* After each iteration, the deliverables will be merged into the Master and tagged with the iteration 0.
* To avoid conflicts on documents, manual versioning is needed in addition to the versioning provided by Google docs (since google docs is limited to 30 days, or 100 versions).
* Major updates to be indicated in the revision. For example - addition of new content, update of critical information and feedback from each iteration.
* For minor updates, the member can indicate the date and username to show latest update from the meeting. For example, 11/18SR: update - would indicated that the update was made by Saloni Rawat on 11/18.

## Code commit guidelines

* Team members to create a new branch before they start working on a new user story.
* Before working on a new user story, team to pull the code from Master before committing any changes.
* Once the branch has been merged with DEV or Master, the branch can either be deleted or rebased from the master.
* At the end of each iteration, all the changes in DEV should be pushed to Master.
* No push to Master in the middle of the iteration.
* All master commits to be handled by Mike Zhong to ensure no conflicts.
* Any adhoc commits to be communicated through slack.
* Adhoc commits should only be made to individual branches. Any share user story to be pushed to DEV, so that it can be used by other team members.
* Capcase to be used for classes.
* Camelcase to be used for Methods
* Each commit should be tagged with a brief description of the update for reference.
  1. Integration and deployment plan
* All team members should start development branching of from Master, to ensure smooth integration
* All developers should be present at the time of integration.
* Once a story is successfully implemented, it should be immediately merged with existing application.
* All team members to perform testing once integration commit is made to ensure no work is lost
* The application will not be deployed to play store

# References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard. )

# Glossary