**CS673 Software Engineering** 

**Team 1 - KnowItAll**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Daniel Makover | Backup Leader | *Daniel Makover* | 2021-11-11 |
| Gunnar Nichols | QA Leader | *Gunnar Nichols* | 2021-11-11 |
| Heli Kolambekar | Requirement Leader | *Heli kolambekar* | 2021-11-11 |
| Haoyi Zhu | Design and Implementation Leader | *Haoyi Zhu* | 2021-11-11 |
| Kun Mo | Team Leader/ Backup Leader | *Kun Mo* | 2021-11-11 |
| Weiye Xu | Configuration Leader | *Weiye Xu* | 2021-11-11 |
| Yuwei Wu | Security Leader | *Yuwei Wu* | 2021-11-11 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **2** | **Daniel Makover** | **2021-10-22** |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_heading=h.3j2qqm3)

[Related Work](#_heading=h.1y810tw)

[Detailed Description](#_heading=h.4i7ojhp)

[Management Plan](#_heading=h.2xcytpi)

[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](#_heading=h.1ci93xb)

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

[Metrics](#_heading=h.3whwml4)

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

[Inspection/Review Process](#_heading=h.2bn6wsx)

[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](#_heading=h.qsh70q)

[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](#_heading=h.3as4poj)

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

[Glossary](#_heading=h.z337ya)

# Overview

KnowItAll is an innovative and interactive trivia web application which allows players to test their knowledge across various categories such as mythology, science, geography, computer science etc. KnowItAll. It allows users to create a player account and manage their profiles. It allows players to post their scores onto a global leaderboard.

The motivation for the proposed project derives from the desire to create a unique trivia web application with real world utility and impact. Our motivation includes the desire to learn software development skills, agile methodologies, software architecture, design principles. The purpose of the proposed software project is to develop software using secure software development practices while utilizing the latest software development tools and technologies.

Online gaming industry has a very high number of active users. It has an ever increasing market share of advertisement revenue and corporate investment. The potential users for the project include seasoned trivia players, beginners and players motivated by lucrative rewards.

# Related Work

Random Trivia Generator is a popular trivia generating web application based on customizing user preferences. However, unlike the proposed project it requires browser web extensions like Guardio to ensure safe browsing to prevent malware attacks from accessing user information . This discourages users from participating in the trivia. The proposed project requires no browser web extensions.

Other trivia platforms such as Trivia Crack have excellent user interfaces but have different features compared to the proposed project. Trivia Crack requires multiple players to answer as many questions as possible in a given amount of time.

Sporcle is another popular quizzing web application which has a high market share amongst all other websites. Sporcle can be used to generate quizzes and share them with friends. The proposed project does not allow users to generate quizzes however users can participate in multiplayer games by sending invites via social media platforms.

Several trivia web applications exist as published github projects which use frameworks such as Flask, Django and Tornado. These projects provide an excellent insight into user preferences, popular features and software architecture. These published github projects have been used as references for the proposed project.

# 3. Proposed High level Requirements

**Functional Requirements**

1. **Essential Features**
2. User authentication - Users can create an account with username, email ID and password. Players can access the game session through a unique username
3. Categories- Users can choose to answer trivia from topics such as geography, TV shows, mythology, computer science, science, art and movies.
4. Quiz - For every question, the player must select the correct answer from four different options within the stipulated time frame. Players score points for every correct answer. The points scored by the player is inversely proportional to the time taken to correctly answer the question. Three incorrect answers cause the player to lose the match. Final player scores are recorded and compared against top scores on the leaderboard.
5. **Desirable Features**
6. The Leaderboard- The top 10 players with the best scores in every category are featured on the leaderboard.
7. In-Game Lifelines- The 50-50 lifeline and skip question lifelines are available to the player during the game session. The 50-50 lifeline disables and removes two wrong answers from a question which gives the player a higher chance of answering correctly. This lifeline can only be used 3 times within a single game session. The Skip Question lifeline enables a user to skip a given question without losing a player life. Points are deducted from the player score whenever this lifeline is used.
8. Player Profile- A user can edit and save the player profile.

c) **Optional Features**

1. Multiple Users - The game session can handle multiple users
2. Password reset and Email Verification- User can reset their password. Email verification will be used to verify user accounts.
3. Ability to restore game session
4. User Feedback- Users will be able to send feedback and submit a rating.

**Nonfunctional Requirements**

**Security Requirements**

1. User Authentication using email
2. Password Strength Settings
3. Backup and Recovery Options

# Management Plan

## Process Model

Our team will work in a semi agile style, drawing on elements from both Extreme Programming and SCRUM. We will have around one or more meetings a week. One meeting will be formal and will focus on planning for the week, while the other meetings will be informal and act as more of a follow-up/Check In. The amount of informal meetings will be determined on an as needed basis. Tasks will be delegated based on priority with Essential tasks coming before Desirable tasks, coming before Optional tasks. Tasks will also be spread across all team members to ensure that every team member makes a meaningful contribution to the application. All deployments will be handled one feature at a time to ensure the applications stability. Additionally, all features will have both unit and features tests to prevent deployment of bugs.

## Objectives and Priorities

Priority 1: Create a fully functional trivia app with all essential features

Priority 2: All team members involved in making meaningful contributions to the application.

Priority 3: Application is deployed with no bugs

Priority 4: Committed code is clean, readable, and understandable by all team members

Priority 5: All desirable features are created and deployed

Priority 6: All optional features are created and deployed

**Risk Management** (need to be updated constantly)

Our main risk is a Technical bottleneck. Due to our team members having a variety of experience and know-how in software development and the technology stack chosen there can arise instances where implementing features seems impossible for some and cause a standstill in the development cycle. We believe that this can be resolved by providing all team members with a basic tutorial of the technology stack. We plan on using the tutorial by CleverProgrmamer along with the official flask tutorial. Therefore, when implementation of some feature seems impossible team members will feel confident enough to know where to turn to for help, be it stack overflow, other team members, or the professor.

Risk Management Sheet Link:

<https://docs.google.com/spreadsheets/d/1OmWn5nxt1Dwp9_uYsjBkKfdiE4cVCgN9_o3gH3qKV2k/edit#gid=0>

| **1**  **Probability**    **5** | **High** | |  | 18 | 8,16 |
| --- | --- | --- | --- | --- | --- |
| **Medium** | |  | 13, 14, 15, 19, 20, 21, 23, 25 | 5, 9 |
| **Low** | |  | 3, 10, 11, 12 | 4, 6, 7, 17, 22, 24, 26, 27 |
| **Risk ID** | | **Low** | **Medium** | **High** |
|  | | **25 Impact x cost 1** | | | |

\*Tip: The number in the matrix means the number of row in the excel sheet of risk management. For example, the risk number 13 means the risk is placed in the 13th row of the risk management excel sheet.

Top 3 priority risks (from high to low):

| **RISK** | **Detailed Plan** | **Execution summary** |
| --- | --- | --- |
| 1. Not familiar with the framework used  (Team members can't contribute work due to lack of programing framework know-how) | Choose a framework that most people know and is easy to learn. Find quick tutorials for team members to get them up to speed | All team members are learning Flask Framework |
| 2. Lack of motivation or responsibility  (required work is not completed) | re-delegate tasks to team members based on motivation and experience | No work has been done as of yet |
| 3. Useless work  (unnecessary work) | delete useless work from code and ensure it's not deployed. Refactor code and redefine features to account for time loss. | No work has been done as of yet |

## Monitoring and Controlling Tools and Mechanisms

We will use the following tools to facilitate group communication and monitor the project progress.

* + 1. Pivotaltracker Link:

https://www.pivotaltracker.com/n/projects/2531648

* + 1. Slack Link:

https://app.slack.com/client/T02CJ0VH7QR/D02CQTTGE4T

* + 1. Github Link:

<https://github.com/BUMETCS673/BUMETCS673A1F21P1>

* + 1. Project Folder: https://drive.google.com/drive/folders/1GTdZs-ZQYDpPYZ7jBpGEXHWRPWOCxaaY
    2. Zoom meeting Link: https://zoom.us/j/93503748719?pwd=VFlxQ1MwVWdtWkNQVFFLRTFBb2lPZz09
    3. Weekly meeting time: Saturday at 1PM

## Timeline (need to be updated at the end of each iteration)

| Iteration | Functional Requirements(Essential/Desirable/Optional)  *(From users point of view)* | Tasks  *(from Devs point of view)* | Estimated/real person hours  *(RANGE)* |
| --- | --- | --- | --- |
| 1 | Basic Single player play(E)  GameOver(E)  Skip Question Lifeline (E)  Question Timer(E)  Leaderboard (D)  Points System(O)  Player Hints (D) | Sample Questions  Internal Game Logic  Database set up  Question Model  Leaderboard Model  Game Logic -Three strike system  Game Logic - hints  Game Lifeline - Skip Question | 14-18  6-10  6-10  6-10  6-10  6-10  10-15  10-15  10-15 |
| 2 | User Profile(E)  Question Categories(E)  Login System (E)  Leaderboard (E)  50/50 Lifeline (E)  Refactor for Security(D)  UI Upgrade (O) | Implement Clock object  User Model  User Authentication  Email verification  Category Model  Game Logic - rewards  CSS & JS refactor  Unit Tests  Feature Tests  Game Lifeline - 50/50 | 10-14  6-10  14-18  10-14  4-10  4-10  14-18  14-18  6-10  10-14 |
| 3 | User Profile(E)  Multiplayer (D)  Restore Game Session(D)  Multiplayer Scoring (D)  Reward Badges(O) | GameSession Model  Game Logic - Scoring  Unit Tests  Feature Tests | 10-14  10-14  8-12  8-12 |

# 

# Quality Assurance Plan

## Metrics

*Metric: What it is. How it is tracked. How it is analyzed and used improve or understand the project*

* + 1. **Product Metrics**

*Lines of Code:* Total lines of code written. This is a metric of capturing how complex the product has become.

*Number of Classes:* Total number of classes written. This is a metric of capturing how complex the product has become.

*Number of Features:* Total number of features planned to be implemented. This is a metric of capturing how complex the product will become.

*Defects*: Total Number of Defects Reported. This will be tracked on Pivotal Tracker. This will allow us to capture how many defects have been reported, and give us an idea of how reliable the completed work has been.

*Defect Fix Rate*: (Total Number of Defects reported as fixed – Total Number of fixes rejected) / (Total Number of Defects reported as fixed + Total Number of new Defects due to fix) \* 100. This will be tracked on Pivotal Tracker. This will allow us to track how effective we are at fixing defects.

*Tests Written:* Total number of tests written. This is a measurement of how many tests have been written

*Tests Passed:* Test Passed / Tests Written. This is a measurement of how well the current code passes the written tests.

* + 1. **Process Metrics**

*User Story Points*: Total Story Points associated with the project. This will be tracked on Pivotal Tracker. This will capture the magnitude of effort we associate with the features needed to complete the project.

*Weekly Story Point Velocity*: Weekly average of story points completed. This will be tracked on Pivotal Tracker. This will allow us to understand the rate at which we implement features and the magnitude of features we can expect to implement if we continue at this rate of productivity.

*Total Man Hours*: Total hours spent working on or preparing for the project. This will be tracked in the Progress Report. This will allow us to track the total effort expended in support of this project.

*Man Hours without Learning*: Hours spent working on the project, not including hours spent learning relevant material required to contribute. This will be tracked in the Progress Report. This will allow us to ensure all members are actively contributing to tasks which bring the project closer to completion.

* + 1. Results: [STD Document](https://docs.google.com/document/d/1ZJ2FKM4mqpSNO6R1EQh0lWgFuvhPQJ-uByH16D7fKxc/edit#)

Standard

* + 1. **Documentation Standard**

All project documentation (SPPP, SDD and Testing) will follow the format established by the Professor’s Templates

Meeting Memo – All group meetings will be recorded in the Meeting\_Memo.docx in the project folder following the established template. The Memo Taker roll will be rotated through all group members.

Progress Report – Each individual’s weekly progress report will be completed no later than class the following week. Project Contributions will be updated by each individual for the end of each iteration. Group Iteration Progress will be detailed by the Requirement Leader and summary of group progress data will be updated by the QA Leader and this will be reviewed by the Group Leader weekly.

* + 1. **Coding Standard**

We will eventually be following the PEP8 Standard, more detailed group standard specifications will be established during iteration 3.

## Inspection/Review Process

Documentation Review

An initial draft of any documentation should be made available to the group two days prior to its due date. Following this draft submission all stakeholders for the documentation should review it and submit comments to the drafter as soon as possible to permit the drafter to address comments for the final submission.

Coding Review

Any code written must be reviewed by someone before being merged into the current iteration’s branch.

## Testing

Unit Testing

Unit Testing will be performed by the developer who wrote the code before it is submitted for peer review. Testing methodology and results will be recorded in the testing report

Feature Testing

Feature testing will be performed by the QA leader and stakeholders established for the given feature immediately following the feature’s integration into the project. Testing methodology and results will be recorded in the testing report.

A separate document about testing results will be linked here for the end of Iteration2

## Defect Management

**Defect Types**

Critical Defect – High Priority

Anything which prevents the primary game from being operational.

Moderate Defect – Medium Priority

Anything which prevents the program from functioning as outlined in the documentation

Small Defect – Low Priority

Any minor issue discovered which prevents a functionality from operating as intended

**Defects tracking**

The QA leader will track the defect status using Github issue tracker after the program baseline is established upon completion of Iteration 1.

**Defect Resolution Process**

The teammate who discovered the defect will create a ticket in Github using the issue template and assign it to the developer responsible for that functionality with the following information:

Classification of severity (Critical, Moderate, Small)

Description and screenshot of the defect

Description of what should happen

Deadline of the fix based on project need

Any novel critical defect should be reported to a project leader immediately. Any novel moderate defects should be discussed during the next group meeting

# Configuration Management Plan

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

## Configuration items and tools

* + 1. GitHub & Pivotal Tracker
    2. Python ---- Language
    3. Flask ---- Framework
    4. Possible to refactor ---

## Change management and branch management

* + 1. Branch:
       1. Master Branch --- Main Branch
       2. Feature Branch
       3. Hotfix Branch
    2. Any code we need to change in GitHub has to be in the new branch
       1. Add feature ---> feature Branch
       2. Bug Fix, Refactor, Optimization ----> Hotfix Branch.
    3. Code commit guidelines
    4. When you need to commit:
       1. Hotfix ---- (Bug fixed)
       2. Add new features:
          1. New feature code does not have bugs that will cause the main program to stop working.
          2. Features running flexible.
       3. Code refactoring or optimization
    5. Git Commit Message Conventions (for example: git Commitizen tool)
       1. Type:
          1. New feature (A new feature)
          2. Bug fix (A bug fix)
          3. Docs (Documentation only changes)
          4. Style (Changes that do not affect the meaning of the code)
          5. Refactor (A code that neither fixes a bug nor adds a feature)
          6. Perf (A Code change that improves performance)
       2. Scope of the change
          1. Component or Files name
       3. Short description of the change.
       4. Long description of the change.(Option)
       5. Any breaking changes. (Yes/No)
          1. Description
          2. Impact
       6. Issues references
    6. Git push:
       1. Before push the file to the branch, double check the files you’re going to push are correct
       2. If you do the rm command, double check you’re doing in the cache or in the local.
    7. Git merge:
       1. The people who are in conflict codes should communicate with each other to deal with the codes.
       2. After the code is completed and checked, merge to the main branch.
    8. Release:
       1. Release by tag.
  1. Integration and deployment plan
     1. Flask ----frontend & backend Framework
     2. MongoDB/mySQL---- database
     3. Vscode/Pycharm --- IDE

# References

Random Trivia Generator: <https://www.randomtriviagenerator.com/#/>

TriviaCrack: <https://www.triviacrack.com/>

Sporcle: <https://www.sporcle.com/>

PEP8 Coding Standard: <https://www.python.org/dev/peps/pep-0008/>

# Glossary