KnowItAll Web App

——Iteration 0 Report

Daniel Makover
Kun Mo
Gunnar Nichols
Heli Kolambekar
Haoyi Zhu
Yuwei Wu
Weiye Xu

Introduction

KnowItAll is an innovative and interactive trivia web application which allows players to test their knowledge across thousands of questions from a variety of topics

Users can participate in single-player or multiplayer games to unlock special player badges and win lucrative rewards

KnowItAll’s global leaderboards reflect the top scores in every category

Product Features

Essential Features

Player Profiles, Categories, vast trivia database, leaderboard, hints

Desirable Features

Special Player Badges, Rewards, Multiplayer Options

Optional Features

Connecting to Social Media Platform

Security Features

Email Authentication, Password Strength Settings, Backup and Recovery Options

Framework: Python Flask

Why Flask?

- Flexibility
- Easy to learn
- More control by the developer

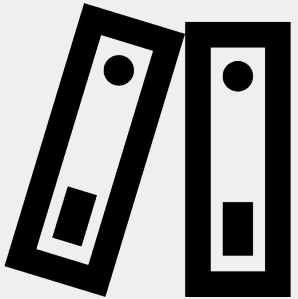
Why Python?

- The one programing language that everyone in the group has basic proficiency with

Flask V.S. Django

	Flask	Django
Type	Light-weight(Micro Framework)	Full-stack framework
Auth	Extension	Built-in user model
Testing	Python's unit test framework & test client & more	Python's unit test framework & test client
Database	Can use many libraries	Built-in ORM
Flexibility	High flexibility	Low flexibility
Learn	Easy to learn	High learning curve

Management Plan



Project Model

Agile, mixing SCRUM and XP, with meetings twice a week. We will utilize the best features of both SCRUM and XP to maximize the product quality and team efficiency

Tasks in a given sprint will be delegated based on priority of importance

All tasks will be spread across team members to ensure that all team members make meaningful contributions to the project.

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

Monitoring and Controlling Tools

TIMELINE

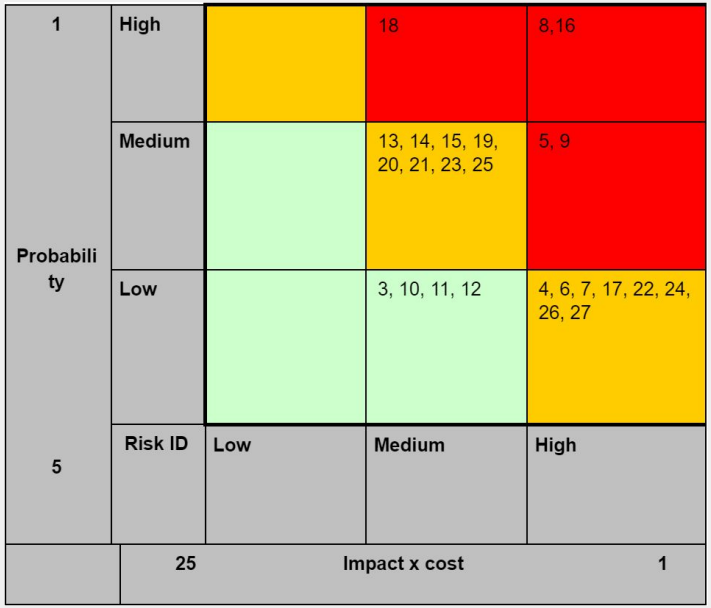
ITERATION 1:
Basic Single Player Game

ITERATION 2:
More features: User profiles/accounts, etc

ITERATION 3:
Multiplayer features & refactoring, single page game(better UI etc)

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) Leaderboard (D) Points System(O)	Sample Questions Internal Game Logic Database set up Question Model Leaderboard Model Game Logic -Three strike system Unit Tests Feature Tests	6-10 6-10 6-10 6-10 6-10 6-10 10-15 10-15
2	Question Timer(E) User Profile(E) Question Categories(E) Player Hints (D) Player Rewards (D) UI Upgrade (O) Reward Badges(O)	Implement Clock object User Model User Authentication Email verification Category Model Game Logic - hints Game Logic - rewards CSS refactor Unit Tests Feature Tests Game Logic - Badges	10-14 6-10 14-18 10-14 4-10 4-10 4-10 14-18 6-10 6-10 6-10
3	Multiplayer (E) Multiplayer Scoring (D) Single Page Application(O)	GameSession Model Game Logic - Scoring Create API Endpoints Implement ReactJS Code Refactor for SPA Unit Tests Feature Tests	10-14 10-14 6-10 14-18 6-10 6-10 6-10

Risk Management



All other risks are highly unlikely

Most Likely Risks

RISK	Detailed Plan	Execution summary
Not familiar with the framework used	Choose 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
Lack of motivation or responsibility	re-deligate tasks to team members based on motivation and experience	No work has been done as of yet
Useless work	delete useless work from code and ensure its not deployed. Refactor code and redefine features to account for time loss.	No work has been done as of yet

Quality and Assurance Plan

Product Metrics

- Complexity: Lines of Code, Number of Classes and Number of Features
  - These three metrics capture the final and current complexity of the product
- Defects: Total Count, Defect Fix Rate
  - These metrics allow us to manage how well our implementation is being realized

Process Metrics

- User Story Points, Weekly Story Point Velocity,
  - These metrics track the total amount of work required and how quickly work gets done
- Total Man Hours and Man Hours without Learning
  - Tracking the contributions of each individual as a whole and on deliverables.

Defect Types

- Critical Defect, Moderate Defect, Small Defect
  - Defect magnitude corresponds to how significantly the bug negatively impacts the games functionality
- Defects will be tracked in Pivotal Tracker

Testing Plan

- Unit Testing
  - Unit Testing will be performed by the individual writing the code before integration
- Feature Testing
  - Feature Testing will be performed by feature stakeholders and reviewed by the QA Lead.

Configuration Management Plan

Configuration items and tools

- Github & Pivotal Tracker
- Flask ---- Frontend & backend framework
  - ---> Refactor to other frameworks
- Python ---- Language
- MongoDB/MySQL ---- database
- Vscode/Pycharm --- IDE

Branch Management

- Master Branch
- Feature Branch ----- add new feature
- HotFix Branch ------ Bug Fix, Refactor, Optimization

Code Commit guidelines

Commit Message Conventions:

- Type (New feature, Bug fix, Docs, Style, Refactor, Perf)
- Scope of the change
- Description of the change.
- Any breaking changes
- Issues references

Git merge:

- The people who are in conflict codes should communicate with each other to deal with the codes.
- After the code is completed and checked, merge to the main branch.

Release:

Tag