CLUE-LESS PROJECT PLAN

October 6, 2016

INTRODUCTION

HailToo is a cloud-based, multiplayer web application which features Clue-like gameplay for 2 to 6 players. Today's entertainment world is riddled with remakes, but HailToo does not fit the mold and introduces a new spin on a classic board game. This instantiation of Clue features a transit theme in it's final implementation where players navigate from neighborhood to neighborhood interrogating suspects as they drive them to their destinations.

Herein lies the document which outlines the project in increments and details what is intended to be delivered at each increment. Beyond implementation we also describe how quality of work will be ensured, the primary responsibilities of each team member, and a target schedule to deliver progress of the project.

The project deliverables will consist of documentation developed by the HailToo team and approved by the customer. Documentation includes design document, requirements document, end-user manual, and executable binaries for the client and server applications. The design document and requirements document will be referenced throughout the development phase to ensure that the product is being built based according to the requirements of the customer. The end-user manual will be delivered with the executable binaries as a reference to how the product should work. There will be one executable binary in the form of a graphical user interface for the client and an application programming interface for the server.

APPROACH

Responsibility	Persons responsible
Overall Project Manager	Christopher T.
Lead Software Developers	Wes W., Christina T.
Quality Assurance Manager	Jameka H.
End-User Documentation Manager	Jameka H.
Requirements Development	Christina T.
Software Architecture	Wes W.

Responsibilities Defined

As Overall Project Manager (PM), Christopher will oversee the progress, and any factors concerning schedules, budgets, and general logistical concerned pertaining to the project. Christopher will demonstrate foresight and problem detection, detect assumptions and resolve conflicts amongst the team, with the ultimate goal of meeting the target deadlines with a quality product.

The task of software development will be tackled by Christina and Wes. These two will translate requirements and specifications into a collection of compute code. They will showcase their technical expertise through the quality, timeliness and thoroughness of the functionality delivered. The software implementation will require the developers to provide solutions for the full stack of the project, from graphical user interface (GUI) to application programming interfaces (APIs), user experience and visual design.

The creation of any software is preceded by the requirements. As the Requirements Development Lead Christina will transpose the desires of our stakeholders into specific, granular statements which are independently implemented and tested.

Software developers will implement code following the architecture provided by the Software Architect, Wes, who will determine structure of the system(s) on a macro-level and make decisions on which technologies (languages/platforms/algorithms/frameworks) to use throughout the project. The architect will make their decisions based on the requirements provided, the time constraints for implementing each feature, and the costs associated with making changes to these choices later down the road.

The Quality Assurance tasks will be overseen by Jameka, who will ensure that developers create testable code, that features function as intended, and the overall program is consistently reliable and available. Jameka will see that each increment of the program is examined and critiqued agnostic to the underlying technologies as development progresses... this tactic will help to continually evaluate the overall behavior of the system. In the effort to guarantee a high quality product Jameka will utilize her familiarity with the project's intentions and implementation to direct the end-user documentation. End-user documentation will be published adjacent to the HailToo code base.

SCOPE

Together we intend to create a web-based, multi-player entertainment experience that will boggle the mind, hopefully. Our team consists of four members with diverse backgrounds, hailing from all over the country and hope to deliver a working, effective game that will impress its users.

The game, in its essence an homage to the board game Clue, introduces a hip new twist to the well-trodden classic via HAIL's expertise – ride sharing! HailToo incorporates a handful of ride-sharers (players), a set of vehicles and the opportunity to solve a tragic hit-and-run mystery. There are 9 interconnected boroughs1, 6 ride-sharing drivers2 , and 9 vehicles3 to consider when solving the mystery of who struck Mr. Boddy, in what borough, and with which vehicle!

CLUE EQUIVALENTS

[1] Boroughs (or neighborhoods) are the Clue-equivalent of rooms.

[2] Ride-sharing drivers (people/actors/characters) are the Clue-equivalent of suspects.

[3] Vehicles are the weapon.

MILESTONE LIST

Target Date	Milestone
2016-10-13	Requirements defined and documented
2016-10-20	Interface specification defined and documented

2016-10-27	Project design completed
2016-11-03	Game implementation meeting skeletal vision
2016-11-17	Game implementation meeting minimal vision
2016-12-08	Game implementation meeting target vision

VISION

Overview

Client - Graphical User Interface

In order to reach more users and to provide the simplest means of playing HailToo, each player will access the game through their web browser. The user interface (UI) will be created through a mix of modern web technologies (e.g. HTML5, JavaScript, CSS, etc), and will initially present the user with a menu which will prompt the options of creating or joining a game. Upon entering a game board should appear and all players' game pieces will be displayed in their appropriate positions. The user will be able to move their game piece on the board at the appropriate time to take action in the game.

Server - Application Programming Interface

While most game logic will be implemented in the client, many of the move validations and state persistence of the game will be implemented on the server. HailToo will tightly interact with an application programming interface (API) which will provide the ability to persist and track game states and their users. This also means that each move made on the game board will be validated by the server for correctness and any attempts to solve the mystery will be checked on the server. The solution to each game will not be accessible by any user, only through an administrative interface will the solution be readable. Each solution will be randomly generated by the server to ensure a neutral playing field for all participants.

Increments/Phases

Skeletal

Basic architecture

- Client
  - Web accessible endpoint
  - Game menu prompt
    - Ability to create a new game
  - Display game board
  - Player's game piece shown on board
    - Ability to move between rooms and hallways as specified by the rules of the game.
- Server
  - Create game endpoint
    - Generate unique game id, persist users participating in game.
  - Validate move endpoint
    - Verify it's the player's turn

Minimal

Minimal implementation should implement all features mentioned in the Skeletal Vision.

- Client

    - Player's game piece shown on board
        - Ability to move between rooms and hallways as specified by the rules of the game.
    - Make an accusation
- Server
    - Validate move endpoint
        - Verify it's the player's turn
    - Check accusation.
    - Notify players when game state has changed
        - Player makes a move

Target

Target implementation should implement all features mentioned in the Minimal Vision.

- Client
    - Make an accusation
        - Move accused player to alleged scene of the crime.
- Server
    - Check accusation
        - Persist location of any moved players.
    - Notify players when game state has changed
        - Other players are moved via accusation.
        - Game is won.

Dream

With infinite time and resources, HailToo would be designed to incorporate bleeding edge virtual reality (VR) user interfaces. In this phase the user would experience the game in a first-person point of view as the driver, interrogating each suspect face-to-face and operating their vehicle in a very life-like manner. Beyond the the interface, users would be able to authenticate through social media outlets, compete against friends and play through the story-mode where they can play as a suspect in others' game.

WORK BREAKDOWN STRUCTURE

ID	WBS	Task Name
1	1	Clue-less software application
2	1.1	Begin project
3	1.1.1	Assemble team
4	1.2	Initialization phase
5	1.2.1	Team Charter document
6	1.3	Planning phase
7	1.3.1	Project Plan document
8	1.3.1.1	Define goals
9	1.3.1.2	Project Plan document draft
10	1.3.1.3	Project Plan document review
11	1.3.1.4	Project Plan document approved
12	1.3.2	Manage requirements
13	1.3.2.1	Requirements gathering
14	1.3.2.2	Requirements document draft
15	1.3.2.3	Requirements document review

16	1.3.2.4	Requirements document approved
17	1.3.3	Interface planning
18	1.3.3.1	Interface document draft
19	1.3.3.2	Interface document review
20	1.3.3.3	Interface document approved
21	1.3.4	Design planning
22	1.3.4.1	Software architecture discussion
23	1.3.4.2	Design document draft
24	1.3.4.3	Design document review
25	1.3.4.4	Design document approved
26	1.4	Implementation phase
27	1.4.1	Skeletal increment
28	1.4.1.1	Basic system architecture
29	1.4.1.2	Primitive UI
30	1.4.1.3	Unit tests
31	1.4.2	Minimal increment
32	1.4.2.1	Interactive functionality
33	1.4.2.2	UI/UX complexity
34	1.4.2.3	Unit tests
35	1.4.2.4	Integration tests
36	1.4.3	Target increment
37	1.4.3.1	Full game play
38	1.4.3.2	Multiplayer support
39	1.4.3.3	Publicly hosted

CHANGE MANAGEMENT PLAN

For HailToo's change management plan, we decided to take some tips from the Department of Transit for the state of Maryland. Our Change Management Plan will show how changes will be managed and controlled by the HailToo team.

Purpose of Change Management:

Our change management plan will be used to establish change that will be proposed by teammates or customers, accepted or denied by our teammates, and monitored and controlled by our project manager. The change management plan will be used to govern changed to the baseline project. The changes HailToo will have to make will be broken down through the structure and requirements of our project. When changes are made, the procedures will govern changes to the baseline schedule and cost of our project. Our change management plan will address the following:

1. Identification of change requests (Change requests will be kept and sorted by importance)
2. Analysis and documentation of the complete impact of requested changes to the scope of the project
3. Approval or rejection of each of the change requests by each teammate
4. Tracking of all changes and updating of project documentation. (Approved Changes)

Procedures

Change Request Procedure

Any HailToo team member, including the project manager, is given the freedom to initiate a change request. To initiate a change, the HailToo member will submit a change request form to the other members of our team using the OneDrive account set up for our project documents. There will be a separate folder, along with notifications set up

for our team members to be alerted when a new change request has been submitted. Once submitted, our team members will have a chance to look over the change request before meeting with the team, once weekly, to discuss all of the change requests that have been submitted by other team members.

Below is our change request form. Once the requestor completes section 1 of the change request form, it will be submitted to the HailToo OneDrive account where it will then be reviewed by the project manager and the rest of the HailToo team. The project manager will record the requests in our change log document which will also be saved in our OneDrive account. The project manager will give the request a change number and all requests will be reviewed in the order of receipt during the weekly HailToo team meeting.

Figure A. Sample Change Request Form.

Team HailToo Change Request Form			
Project Title: HailToo Clue Game		**Project Number:** 000001	
Project Manager: Christopher T.			
Section 1: Change Request			
Requestor Name: **Requestor Phone:**	**Date of Request:**	**Change Request Number:** *Supplied by (PM)*	
Item to be Changed:		**Priority:**	
Description of Change:			
Estimated Cost & Time:			
Section 2: Change Evaluation			
Evaluated by: What is Affect:		**Work Required:**	
Impact to Cost, Schedule, Scope, Quality, and Risk:			
Section 3: Change Resolution			
Accepted Rejected	**Approved by (Print):**	**Signature:**	**Date:**
Comments:			
Section 4: Change Tracking			
Completion Date	**Completed by (Print):**	**Signature:**	**Date:**
My signature above indicates that the project documentation has been updated to accurately and comprehensively reflect the approved changes.			

Change Analysis Procedures

The team will analyze and assess the impact of each proposed change during our weekly team meeting. The team will take into account what impact each change will have on the baseline scope of the project and what kind

of hit the team's schedule and budget will take for each change made to the baseline project. Once each change is evaluated, the team will go over the impact that change will have on the project before accepting or denying the request. The project manager will then assign a team member to complete section 2 of the change request form. Section 2 will detail the work to complete and the change and impact that change will have on the project and project deliverables.

Change Request Approval Process

When evaluating a change request, the team will make a group decision as to whether or not a change request is viable for the scope of the project. The impact of the change will be recorded by one of the team members. The project manager will pass the decision of the change request onto the HailToo team. If a decision cannot be made in our weekly meeting, we will allow the team to think of the change request and any possible changes before reconvening the following week at our next meeting for reevaluation. Once a decision is made, the team will complete section 3 of the change request form. The form will then be returned to the project manager for documentation.

If approved, the Project Manager will update the project documentation in the OneDrive account and the group will review the changes made to the documentation. Corresponding modifications will be made in appropriate documents in the OneDrive account and each team member will be notified of the changes so they can review and approve the said changes. If the approved change affects schedule, scope or cost, the project term and due date may be negotiated by the team. If the change is rejected, the project manager will update the change control log.

Change Tracking (maintain a master log)

Our Project Manager will be in charge of maintaining a master log of all the change requests and the resolutions of said change requests. The Project Manager will keep a Change Control Log. A sample of the change control log is shown below in Figure B. Our Change Control Log will be kept in the OneDrive account.

If a change is approved, the Project Manager will complete Section 4 of the submitted Change Request Form to show it was completed and all project documents have been updated in the OneDrive account.

HailToo project changes will carry an urgency requirement for each change request. We will categorize changes in the Change Control Log with an urgency from 1-3. Urgency 1 means that it is not an urgent requirement and has little impact on the coding of the game. Since our project will be coded modularly, we will be able to add and remove features fairly seamlessly. For example, if someone put in a request to add more players to the game, it would get an urgency of '1;' however, if someone put in a request to move the completion date back and complete the game sooner, it would be listed as urgency of '3.'

Figure B. Small snippet of Change Control Log (Maintained by HailToo Project Manager, Chris T.)

Project Change Control Log							
Project Title: *HailToo Clue Game*				***Project Number:*** *0000001*			
Project Manager: *Christopher T.*							
Change Number	***Description of Change***	***Priority***	***Date Requested***	***Requested By***	***Status (Evaluating, Pending, Approved, Rejected)***	***Date Resolved***	***Resolution/Comments***
1	*Add More Players to Game*	*1*	*10/10/16*	*HailToo*	*Pending*	*TBD*	*TBD*
2							
3							

COMMUNICATIONS MANAGEMENT PLAN

All members of the HailToo team will meet twice weekly (Monday and Wednesday) to inform team members of current status of project milestones. During these meetings any issues or items of concern will be discussed. The HailToo Project Manager, SW architect, and Lead Developers will meet with the customer and all stakeholders every Thursday to update on status, request clarification of any requirement, and/or seek approval to derive from the approved requirement or design documents. Any issues that arise that may impact schedule, budget, or requirements must be discussed. In this, and any other, case, the HailToo team or the customer may schedule an urgent, out-of-cycle meeting that must be held within 48 hours of request.

COST MANAGEMENT PLAN

Costs for development will be held to a minimum through the use of open source tools and student-licensed software. HailToo will be hosted in a cloud environment (e.g. Amazon Web Services, Digital Ocean) and agree to submit upwards of twenty dollars towards hosting costs over the course of the project schedule (through December 8th, 2016). All team members have agreed to forfeit all potential compensation for work on HailToo to further fund hosting and pizza costs when necessary (HailToo is valued at approximately zero dollars at the creation of this document).

PROCUREMENT PLAN

N/A

QUALITY MANAGEMENT PLAN

Project implementation will heavily rely on the standards and policies set in the quality plan for optimal development practices and fortified milestone achievement. The project quality plan will be defined by quality assurance of code delivery, concrete testing procedures, and exceptional configuration management standards. Protocol and conventions for these provisions are stated and defined below.

Quality Assurance

Quality assurance (QA) will be implemented into development, test, and configuration management principles. Software QA practices will ensure that all software has been properly tested and all requirements have been met, and establish the reliability and dependability of deployed software. Software performance must be evaluated to satisfy the needs of everyday use. Code must be well-structured and understandable, and all programming documentation standards will be followed during the development process. QA procedures will extend into configuration management and testing approaches which are thoroughly defined in the proceeding sections.

Testing Approach

Embedded testing will cover various principles from unit, acceptance, functional, performance, and security testing practices. The Quality Assurance Manager will comprise a team of testers to conduct end-to-end testing of the client application and server to include functionality and performance. All bugs and/or unintended behaviors will be documented and labeled by severity level 1-3 (1 = Minor (Enhancement); 2 = Major (Workaround or quick fix has been identified); 3 = Critical (Functionality or performance issue that will prevent the product from working).)

Configuration Management

The team's configuration management approach will ensure that all members have access to the most up-to-date versions of code and project documents in a controlled environment. All members will have 24/7 access to project edits and additions made in real-time in order to compile and link components in a synchronized manner for successful system integration. Configuration management protocol is established via version management, system integration, problem tracking, and release management (see below descriptions).

Version Management

Version management will be implemented to coordinate development by multiple programmers. The team will be using GitHub, a Git repository hosting service built with distributed version control and source code management policies; these features will prevent one developer from overwriting code that has been submitted to the system by another developer to enforce versioning synchronicity throughout the development lifecycle.

System Integration

System integration will assist in directing which component versions will be assigned to each version of the system. The team will be utilizing Gradle, a build automation tool complete with a Groovy-based domain-specific language to determine project configuration. Automated builds will be accomplished by compiling and linking all required components, with build fail notifications should conflicting components or other inconsistencies appear in code submitted to the system.

Problem Tracking

Problem tracking will enable the team to report problems, view the status of said problems, see who is working individual issues, and view both in-work and resolved tickets. Although the team will not be utilizing an official, commercial-product issue tracking system, there will be a designated project manager (also acting as a scrum master) in charge of delegating initial tasks to team members. Developers have the opportunity to include comments, concerns, and miscellaneous notes (bug reports, bug fixes, quality and efficiency edits, etc.) in code commit actions and issue tracking documentation that will be reviewed and analyzed by the project manner on a continual basis. The project manager will then assign and add/remove tasks from the development queue as necessary, and will monitor each task by pinning status marks to each ticket (not started, in work, in test, etc.) for complete system observation and investigation. The issue tracking document will be available to and editable by all team members 24/7.

Release Management

Release management will be based on system integration procedures and proper planning of release functionality and software organization for distribution. Versioning and functional system capabilities will be thoroughly assessed and resolved before each code delivery to ensure the current state of the product meets the customer's design standards and desired level of performance at the current stage in development.

RISK MANAGEMENT PLAN

Schedule delay due to coding issues, requirement delay, design change.

The impact of a design change could includes schedule delay, increase in cost. To mitigate changes to the design, the HailToo team will thoroughly review the design to ensure that all things have been considered. After reviewing with the team, the HailToo team will review the design with the customer. During these 2 reviews, any issues that arose will be addressed, reworked, and reviewed again with the team. This cycle will continue until the design is agreed upon with little to no room for error.

Software versioning impacts? Browser upgrades? Java versions?

In the instance of a conflict amongst the HailToo team, resolution will be determined via a majority vote. The Project Manager and Software Architect will have 2 votes each while the remaining team members will have 1 vote each. The Software Architect will serve as the ultimate authoritative decision maker.

STAFFING MANAGEMENT PLAN

Staffing roles have been established and assigned to each member of the team. Staff positions include an overall project manager, lead software developers, a quality assurance lead, an end-user documentation manager, a requirements development lead, and a software architecture lead.

The overall project manner (PM) is responsible for complete oversight of the project, serving as both a scrum master and technical delivery manager. The PM is accountable for final decisions made with regards to role assignments and all other areas in the entire realm of the project plan. Lead software developers are responsible for writing, documenting, and deploying all of the code that is written. Developers establish development tasks necessary for completion during each delivery phase, demonstrate exceptional code review practices, and follow all deployment procedures outlined in the configuration management and delivery plans. The quality assurance lead abides by and endorses the quality management plan throughout the development lifecycle and continuous post-delivery quality assurance. The end-user documentation manager is responsible for constructing release notes (sets of instructions and guidelines for game play and other application operations and functionality changes/updates) with every deliverable. Documentation should be clear, concise, and designed with a clientele mindset. The requirements development (RD) lead is tasked with creating both functional and non-functional system requirements that are both applicable and testable. Base requirements set by the requirements manager (via client requests) are further defined by requirement sub-components. The RD must both establish initial requirements and manage changing requirements throughout the development process. The software architecture (SA) lead is the designated system and application design lead. The SA handles development tool product procurement in conjunction with the PM, and implements all system structure and design; this includes development and deployment tools used, back-end and client-side frameworks, code design patterns, and UI feature models.

Note that each of these roles are distinguished as "lead" positions— this means that a majority of the work that falls under these job descriptions will be done by those listed at the leads. This does not mean, however, that the responsibilities of these roles cannot be taken on by someone else in the event that the team becomes temporarily or permanently shorthanded. Each team member currently averages 2 lead roles, and these can be adjusted or reassigned as deemed necessary throughout project development. Reasoning for role reassignment will be based on varied availability of group members (due to temporary absence), or a teammate's desire to take on a specific task tied to a role in which they are not a lead. This may occur due to varying levels of expertise among different disciplines, and team members may be asked to assist with work across different roles.

Roles and responsibilities will be established at the start of delivery phases, and re-evaluated upon completion of milestone deliveries to asses and reconsider any changes to role assignment deemed necessary. Metrics will be maintained to track progress, efficiency, and effectiveness of team member's performance. An open

discussion will be held upon milestone completion to give the team the opportunity to discuss how each member feels about their individual contributions and make note of any room for improvement or suggested shuffling of team roles.

DELIVERY PLAN

The project deliverables, including executable server binary, end-user manual, requirements document, design document, and troubleshooting guide will be copied to a CD and hand-carried to the customer. At delivery, a demonstration will be provided to the customer. Additionally, HailToo offers support in the intial configuration of the Clue-Less server. HailToo will provide lifetime troubleshooting support. However, in the event of code changes or revisions after the closing of the project, an additional fee will apply to make code modifications or updates.

APPROVAL AND AUTHORITY TO PROCEED

We approve the project as described above, and authorize the team to proceed.

Name	Date
Chris Tarvin	10/06/2016
Jameka Hicks	10/06/2016
Christina Torpey	10/06/2016
Wes Walker	10/06/2016

REVISIONS

Version	Primary Author(s)	Description of Version	Date Completed
Draft	TBD	Initial draft created for distribution and review comments	Sept 27, 2016
Preliminary	TBD	Second draft incorporating initial review comments, distributed for final review	Oct 3, 2016
Final	TBD	First complete draft, which is placed under change control	Oct 6, 2016
Revision 1	TBD	Revised draft, revised according to the change control process and maintained under change control	TBD
Revision 2	TBD	Revised draft, revised according to the change control process and maintained under change control	TBD
etc.	TBD	TBD	TBD