

# ***Project: Red Pony Plan***

## **Introduction**

### **Project Scope**

Project: Red Pony will consist of a framework (The Astral Framework) and a social network application that uses the framework to show its applicability. The Astral Framework aims to give physical-world restrictions, like time and place, to files uploaded in a digital environment. <Application> will use the framework and show its applicability.

### **Framework Inputs**

The framework requires the following inputs:

- For retrieving:
  - Location: User's current location
  - Time: Current time at user's location
  - ID: User ID
- For uploading:
  - Location: location restriction of file
  - Radius: radius for file accessibility
  - Time: time restriction for file
  - Data: Data to be stored

### **Application Inputs**

To use the application, the user must provide various inputs:

- In order to access this application:
  - Create an account providing a username, email, and a password
  - Use a third party account to log in
- In order to use this application:
  - Users must give the app permission to use location services
  - To upload files users must also give the app permission to access files and photos stored on the phone.
  - Users can optionally give permission to use the camera.
  - Users can optionally allow push notifications

### **Framework Functionalities**

In order for the framework will be responsible for:

- File Management

- Manage database of all files
- Account Management
  - Manage database of all users and their information

## Application Functionalities

- Location Services
  - Constantly track user location
  - Send location to server for file retrieval
- File Sharing
  - Allow the user to upload files
  - Allow user to view nearby files
  - Allow user to create rooms
  - Allow user to join rooms
    - Upload and download files to rooms
    - Comment section in rooms
- User Interface
  - Provide an interface for account creation/login
  - Provide an interface for file sharing
    - Uploading a new file
    - Setting restrictions
    - Downloading available files
  - View available files on a map
  - Provide an interface for user settings

## Major Software Functions

- *File and Account Management*
  - MongoDB: User accounts and uploaded files will be stored and managed with MongoDB, a noSQL cloud-hosted database as a service tool.
- *Hosting*
  - Google Cloud: Application backend will be hosted on Google Cloud.
- *User Interface*
  - Ionic: A free, open-source, and cross-platform framework for developing mobile applications with web technologies like AngularJS, Javascript, Typescript, and HTML5.
- *Location Services*
  - Ionic-Native Geolocation: Ionic comes with a plugin for geolocation that provides latitude, longitude, altitude (and its accuracy), heading, and speed.

## **Performance/Behavior Issues**

Performance will be directly related to internet connection.

Android Users' devices must be updated to an API level equal to or higher than 21 while iOS device users should have a device supporting a version of iOS 9.3 or higher.

## **Management and Technical Constraints**

### Deadlines

#### **Fall 2017:**

- Date - 10/2/2017
  - Project Plan
- Date - 10/9/2017
  - UML Diagram
- Date - 10/16/2017
  - Flowchart
- Date - 11/13/2017
  - Use Cases
- Date - 11/13/2017
  - Test Plan
- Date - 11/27/2017
  - User Interface Layout
- Date - 12/4/2017
  - User Manual
- Date - 12/4/2017
  - Presentation and Application Tutorial

#### **Spring 2018:**

- Date - 2/19/2018
  - Geolocation services
- Date - 2/26/2018
  - File Sharing
- Date - 3/5/2018
  - User Accounts
- Date - 3/19/2018
  - Finished framework
- Date - 4/16/2018
  - User Interface
- Date - 4/30/2018
  - Finished application
- Date - 5/7/2018

- Presentation

The design and implementation will be completed according to the prototyping model in *Fig 1*.

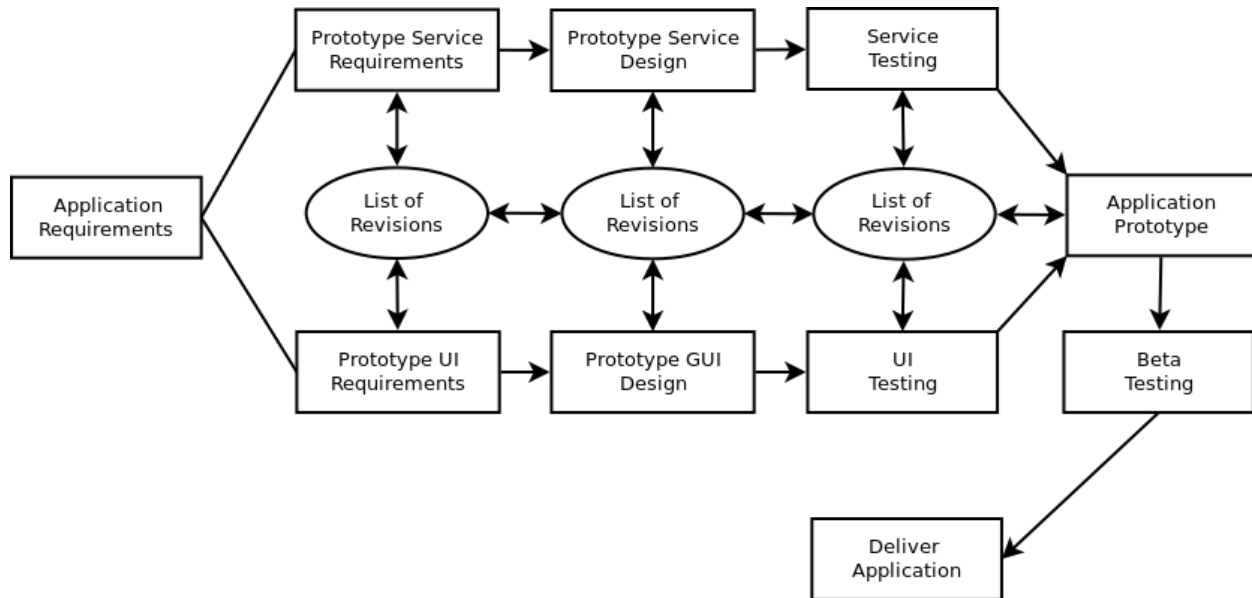


Fig 1: prototype model

## Project Estimates

### Project Resources

The team in charge of creating this application is comprised of 5 computer science students. Each student will be required to assume multiple roles:

#### *Required Staff*

- UI Designer
- Security and Privacy Assurers
- Programmer/Coder
- Graphic Designer
- Quality Assurance Management
- Software Testing
- System Designer
- Documenter
- Database Administrator

## Risk Management

### Project Risks

- Late software delivery
- Inexperience with development tools
- Technology will not meet expectations
- Deviation from software engineering standards
- Poor comments in code

### Risk Table

Risks	Probability	Impact
Late software delivery	70%	1
Inexperience with development tools	70%	2
Technology will not meet expectations	20%	2
Deviation from software engineering standards	15%	3
Poor comments in code	10%	4

### Overview of Risk Mitigation, Monitoring, and Management (RMMM)

Risk will be mitigated through thorough testing and documentation. If any risk is found, a strong method will be determined to find the best course of action to take if the risk occurs. The Requirements Specification and the System Specification will be analyzed to find any risk that may happen as a cause of this software. We will create predetermined paths and directions to follow when attempting to manage a risk.

### Project Schedule

#### Project Task Set

##### *Process Model*

The design and implementation will be completed according the prototyping model in *Fig 1*.

##### *Framework Activities*

- Planning and Design
- Risk Analysis
- Programming
- Testing
- Client Feedback

##### *Task Set*

- Framework construction
- User Interface Construction

- Testing
- User manual construction
- Application construction

## **List of Deliverables (In order of Importance from the Top)**

### *Documentation*

Project Plan  
Flowchart and UML Diagram  
Use Cases  
Test Plan  
User manual and User Interface Layout

### *Code*

Framework  
User Interface  
Online account profiles  
Application

## **Functional Decomposition**

### *Framework*

- Account creation
- File uploading
- File downloading
- File querying
- User querying

### *User interface*

- Account creation interface
- File query interface
- Map interface

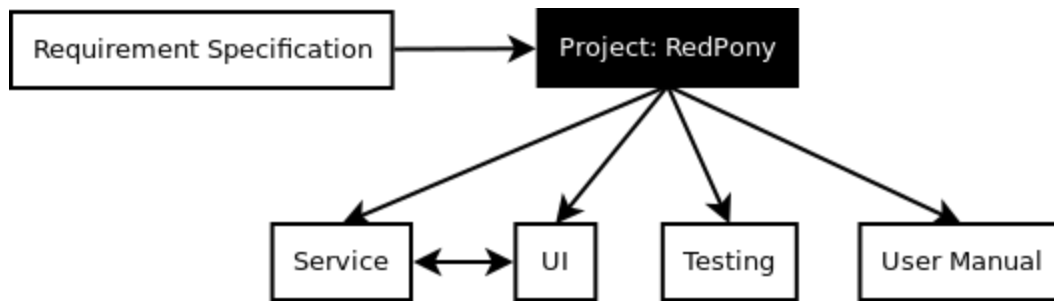
### *Testing Task Breakdown*

- Test file querying
- Test account creation
- UI user-friendliness
- UI aesthetic (smoothness of transitions, appealing design, etc.)

### *User Manual Task Breakdown*

- Interface manual construction
- OS requirements construction
- FAQ construction
- Manual construction

## Task Network

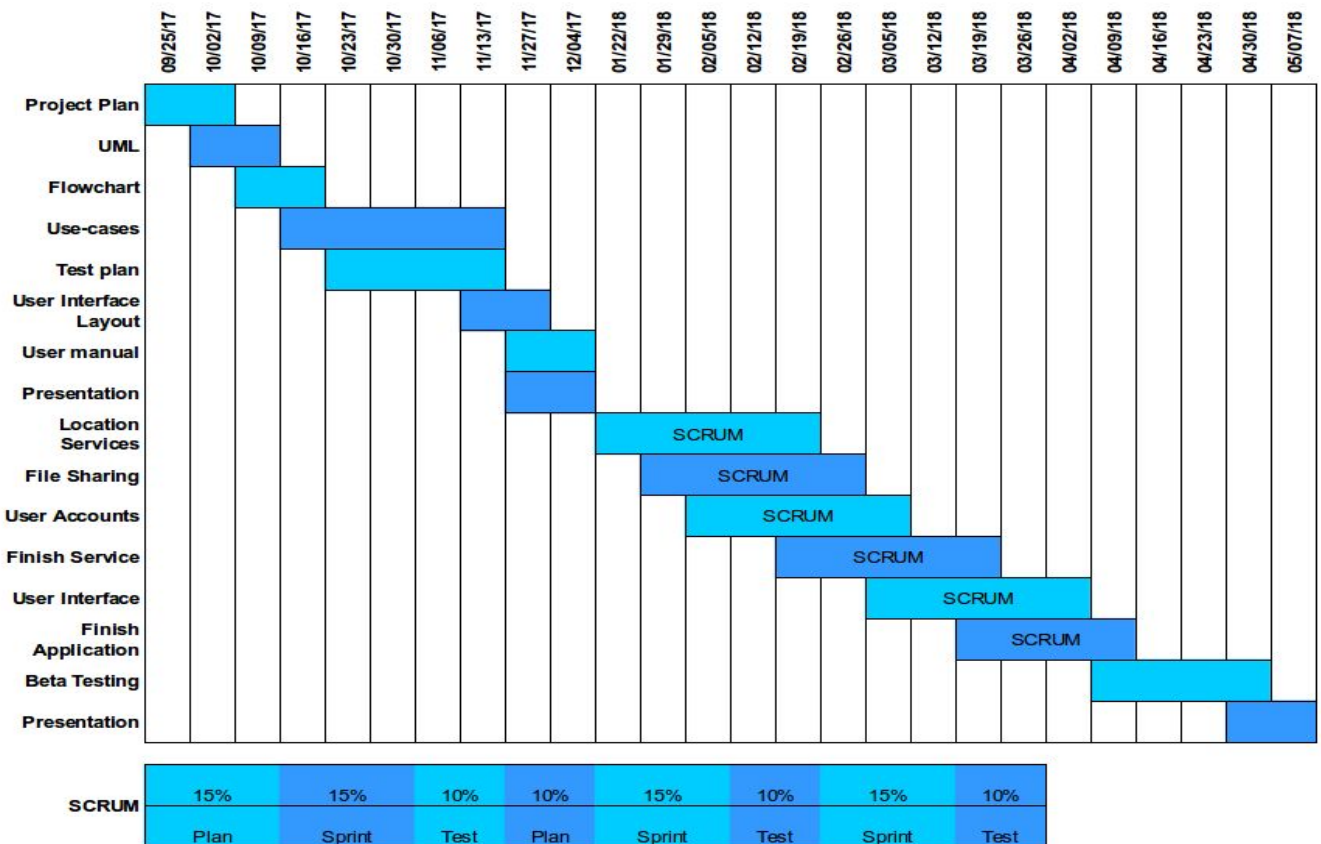


## Software Methodology

### Scrum

- Plan: agree on goals for sprint
  - Weekly meetings
- Sprint: Coding sprint with well defined goals and requirements
- Test: Test code from sprinting phase and use results for next plan/sprint phase

## Timeline Chart (Estimate)



## Team Structure

The Project: Red Pony team will use the Egoless Democratic Model for team structuring:

- The Egoless Democratic Model states that every goal is set by a consensus between team members. Every member's input will be taken to account for every major decision and group leadership will rotate depending on the specific skill set of individual group members.

### Role Definitions

*Daniel Olivera*

**Software Tester:**

Runs and validates the test cases to ensure functionality.

**Quality Assurance Management:**

Confirms the application works as described and is functionally complete.

**Programmer/Coder:**

Helps in the development of the application through coding

**Documentation:**

Updates documentation throughout the life of the project.

*Daniel Kim*

**User Interface Designer:**

Oversees the creation of the UI. In charge of design and implementation.

**Programmer/Coder:**

Helps in the development of the application through coding.

**Documentation:**

Updates documentation throughout the life of the project.

*Jared Coleman*

**System Designer:**

Creates the backend of the platform on which the application will run. Ensures all parts of the framework and application backend work as a single system.

**Programmer/Coder:**

Helps in the development of the application through coding

**Documentation:**

Updates documentation throughout the life of the project.

*Mario Lopez*

**Database Management:**

Creates and Maintains the database used to store user's files and makes sure it integrates seamlessly with the application.



**Programmer/Coder:**

Helps in the development of the application through coding

**Documentation:**

Updates documentation throughout the life of the project.

*Viet Le*

**Security and Privacy Administrator:**

Handles all the security and privacy concerns with the framework and application.

Ensures encryption protocols are up to industry standards.

**Programmer/Coder:**

Helps in the development of the application through coding

**Documentation:**

Updates documentation throughout the life of the project.

**Management Reporting and Communication***Mechanisms for Progress Reporting*

Progress reporting will be done through the use of GitHub and its notification system along with the use of Slack and its tools.

*Mechanisms for Intra/Inter Team Communication*

Team members will meet up twice a week following class lectures to ask any questions and discuss the project amongst each other. Extra meetings will be conducted depending on the importance of the task the team is currently handling at the moment. Outside of face-to-face meetings, team members will use the Slack software tool to communicate to each other about anything pertaining to the project.

**Tracking and Control Mechanisms****Quality Assurance and Control**

Team members will continuously monitor each other's work to ensure that the product does not deviate far from the original design specifications. Any deviations that occur and are spotted will be brought to the attention of the involved team members. Additionally, testing will be done throughout the development stage to recognize any defects in the design or implementation.

**Change Management and Control**

Management will be controlled through the use of GitHub and its version control management system. Each branch to be merged into the origin repository will have to be

confirmed by multiple or all teammates depending on the importance of the item and its impact on the software. Team members will fetch and merge in the changes from another user's repository to thoroughly test the implementation before the addition into the server's main branch. The goal is to minimize unnecessary backtracking so that our product can be developed in a timely manner.