**PubChat**

HBB Enterprise

Benjamin Hapip

Adam Beyer

Aditya Bhasin

**Table of Contents**

Section Page

[1. Project Vision and Objectives 1](#_Toc433722625)

[1.1 Project Scope and Vision 1](#_Toc433722626)

[1.2 Project Goals and Objectives 1](#_Toc433722627)

[2. Project Planning 2](#_Toc433722628)

[2.1 Project Lifecycle 2](#_Toc433722629)

[2.2 Project Setup 2](#_Toc433722630)

[2.3 Stakeholders 2](#_Toc433722631)

[2.4 Project Resources 3](#_Toc433722632)

[2.5 Assumptions 3](#_Toc433722633)

[3. Project Tracking 4](#_Toc433722634)

[3.1 Tracking 4](#_Toc433722635)

[3.2 Communication Plan 4](#_Toc433722636)

[3.3 Deliverables 5](#_Toc433722637)

[3.4 Project Metrics 6](#_Toc433722638)

[4. Requirements (User Stories) 7](#_Toc433722639)

[4.1 Overall Description 7](#_Toc433722640)

[4.2 Users and Roles 7](#_Toc433722641)

[4.3 Use Case Diagrams 8](#_Toc433722642)

[4.4 User Stories (Requirements) 8](#_Toc433722643)

[4.5 Constraints and Limitations 13](#_Toc433722644)

[5. Design 14](#_Toc433722645)

[5.1 Introduction 14](#_Toc433722646)

[5.2 Scope 14](#_Toc433722647)

[5.3 High-Level Component Design 14](#_Toc433722648)

[5.4 Class Diagram 15](#_Toc433722649)

[5.5 Activity Diagrams 16](#_Toc433722650)

[5.6 Sequence Diagram 17](#_Toc433722651)

[5.6 Alternative Designs and Design Rationale 17](#_Toc433722652)

[5.7 Data Architecture 18](#_Toc433722653)

[6. User Interface 19](#_Toc433722654)

[6.1 UI Description 19](#_Toc433722655)

[6.2 UI Mockup 19](#_Toc433722656)

[7. Project Closure 20](#_Toc433722657)

[7.1 Goals / Vision 20](#_Toc433722658)

[7.2 Delivered Solution 20](#_Toc433722659)

[7.3 Remaining Work 20](#_Toc433722660)

[8. Definitions and Acronyms 21](#_Toc433722661)

# 1. Project Vision and Objectives

## 1.1 Project Scope and Vision

## PubChat stems from the crude idea of stranger communication. This application will allow users to communicate anonymously with other users in a certain radius. The goal of this app is to get people to share their thoughts with the confidence of being anonymous. The initial scope is a younger crowd with android based smartphones.

## 1.2 Project Goals and Objectives

|  |  |
| --- | --- |
| **#** | **Goal or Objective** |
| 1 | Chat with others within a couple of meters. |
| 2 | Make the system easy to support – provide good documentation, configuration/build files, administrator’s manual |
| 3 | Make the system very easy to use – users would agree that minimal to no training is needed |
| 4 | Design an early version that we distribute to college students, mall goers, and small restraunts. |
| 5 | Make the system accessible and owned by major companies. |

# 2. Project Planning

## 2.1 Project Lifecycle

The team will use an agile approach. Our team will gather requirements and create a high-level development plan at the onset of the project and then implement the gathered requirements over three iterations. The team will follow a SCRUM-like approach with an emphasis on frequent meetings and collaboration.

## 2.2 Project Setup

|  |  |
| --- | --- |
| **#** | **Decision Description** |
| 1 | Java vs. C# Android Studio vs. Xamarin, Git |
| 2 | A phone needs to host this locally |

## 2.3 Stakeholders

|  |  |
| --- | --- |
| **Stakeholder** | **Role** |
| Person A | Matt Kukla |
| Person B | Ben Hapip |
| Person C | Aditya Bhasin |
| Person D | Adam Beyer |

## 2.4 Project Resources

|  |  |  |
| --- | --- | --- |
| **Resource** | **Resource Description** | **Quantity** |
| Database Server | A database server might want to be considered | 1 |
| Project Team | Our team of students who will be the primary developers of the project. | 3 |
| Mac Workstation | An OS X workstation with X Code for developing the OS X version of the software. | 1 |
| Android Phone | An Android phone to be used as test hardware for the mobile version of the software. | 2 |

## 2.5 Assumptions

|  |  |
| --- | --- |
| **#** | **Assumption** |
| A1 | The team will be able to meet face to face once a week. |
| A3 | Team members will be able to familiarize themselves with Android Studio and or Xamarin. |
| A4 | Team will need to be assigned tasks, and will have consequences if the tasks are not complete within allotted time. |

3. Project Tracking

## 3.1 Tracking

|  |  |  |
| --- | --- | --- |
| **Information** | **Description** | **Link** |
| Code Storage | Project code will be stored in a GitHub repository. | [Link](https://github.com/Pub-Chat/PubChat) |
| Bug Tracking | Bug tracking will be done in the GitHub organization. | [Link](https://github.com/Pub-Chat) |
| Project Documents and Assignments | Specification and design documents, etc. will be stored in our GitHub repository. | [Link](https://github.com/Pub-Chat/PubChat) |
| Regression Testing | Regression testing will use JUnit unit tests and Jenkins. | Not available |

## 3.2 Communication Plan

Regularly Scheduled Meetings

|  |  |  |
| --- | --- | --- |
| Meeting Type | Frequency/Schedule | Who Attends |
|  |  |  |
| Team Meeting | Weekly | Project team |
| Sprint Planning Meeting | Start of each sprint | Project team |
| Sprint Retrospective Meeting | End of each sprint | Project team |
| Sprint Review Meeting | End of each sprint | Project team |

Information To Be Shared Within Our Group

|  |  |  |  |
| --- | --- | --- | --- |
| Who? | What Information? | When? | How? |
| Project team | Task assignments & General scrum information | Weekly | Team meetings, listing in Project Specification. |
|  |  |  |  |

Information To Be Provided To Other Groups

|  |  |  |  |
| --- | --- | --- | --- |
| Who? | What Information? | When? | How? |
| Currently no one to report to |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 3.3 Deliverables

|  |  |
| --- | --- |
| **#** | **Deliverable** |
| 1 | Code |
| 2 | Test and test results |
| 3 | Build process documents***(if any)*** |
| 4 | Install process documents***(if any)*** |
| 5 | Administrator or user manual***(if any)*** |
| 6 | Postmortem document |

## 3.4 Project Metrics

|  |  |  |
| --- | --- | --- |
| Metric | Frequency | Location |
| Estimated User Story Points | Per Sprint at the start of each sprint | At the beginning of Individual Sprint  (Section 4.5) |
| Actual User Story Points Completed (Velocity) | Per Sprint at the start of each sprint | At the beginning of Individual Sprint  (Section 4.5) |

# 4. Requirements (User Stories)

## 4.1 Overall Description

**<< Provide a more detailed, two to three paragraph description of the project. This description may include more technical details to describe the purpose of the project.**

This goal of this project is to design a mobile application that will give the user the ability to instant message with one or more people within a set geographic radius. >>**Needs more still. Just a start**

## 4.2 Users and Roles

<< Provide a list and description of the different types of users or roles within the system. This may include different classes of users, such as administrator, instructor, student, etc. This list may also include autonomous agents that interact with the system as well. These may include users (or personas) that are used as part of any user stories produced for the project. >>

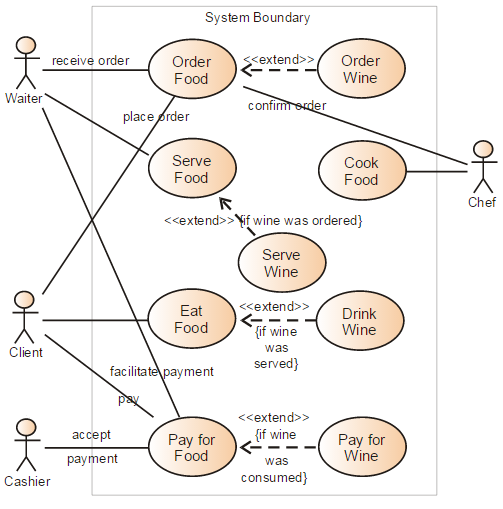
**Example:**

|  |  |
| --- | --- |
| **User** | **Description** |
| Developer | A member of the development team that is tasked with writing code to accomplish user stories. In addition to this, a developer is responsible for documentation, team meetings, design choices, etc… |
| Microsoft Program Manager | A manager at Microsoft who is working on developing the ERP application who will be making design decisions based on the data analysis. |
| PubChat User | An end user of the PubChat application. The user will be logging in to their account and messaging people in a multitude of chatrooms. |

## 4.3 Use Case Diagrams

<< Provide any use-case diagrams that are being used as part of the project. Uniquely label each use case so that if necessary it is easy to reference from other parts of the document. >>

**Example:** Restaurant system



## 4.4 User Stories (Requirements)

<< This section lists the user stories for the project, when they were added, and information about which of these user stories the team has committed to complete (C), stories that will be targeted but not guaranteed to be completed (T), and those which the team will not commit to (NC).

It is preferred to have user stories written in the form - *As an [actor] I want [action] so that [achievement].* So, for example: *As a Bison Tracker member, I want to set different privacy levels on my photos, so I can control who sees which of my photos.*

The **ID** column provides a high-level ID for each user story. This is useful for generating acceptance criteria IDs which can be easily associated with a particular user story. The **Added** column lists when the user story was added to the project as it is possible that not all requirements will be elicited or available at the project’s onset. The **Description** column gives a description of the user story. The **Status** column indicates whether these user stories are Committed (i.e. C will be completed by the team), Targeted (i.e. T will be completed if the team has time after all committed requirements are completed), or Not Committed (i.e. NC will not be completed by the team). The **Story Points** indicate how much work is associated with each story, **Actual Equivalent Story Points** is the actual story points a user story took at the end of the sprint as compared to the estimated in the beginning of sprint. This helps to improve the estimation techniques for future projects. **% Completed** indicates how much work is completed so far for each user story.

Story points are assigned at the start of each sprint for those user stories that are assigned to that sprint. The default is to use values of 1, 2, 3, 5, 8, 13, and 21. Anything bigger than 21 should be made into multiple stories. Each team should benchmark their story points and indicate what was used as the benchmark. For example, in an online bidding system, everyone in the group agrees that the *“Search for an item”* feature is 2 story points, so the points for all other stories are allocated in relation to that benchmark.

|  |  |  |
| --- | --- | --- |
| **ID** | **Feature name** | **Story points** |
| 5 | Register on the site | 3 |
| 4 | Place an item up for bid | 3 |
| 10 | Bid on an item | 3 |
| 17 | Auction engine | 8 |
| 13 | Search for an item | 2 |
| 16 | Purchase an item immediately | 2 |

**Acceptance Criteria** should define the boundaries of a user story and are used to confirm when a story is completed and working as intended. The **ID** column specifies the ID for the acceptance criteria with respect to the corresponding user story ID. The **Description** column specifies the acceptance criteria formally. The **Verification** column provides information about how the acceptance criteria will be tested.

Remember that not all user stories are necessarily functional. User stories can also specify the need for security (e.g. who can and cannot use a system), globalization (e.g. descriptions and other strings will not be hard coded so that the software can be translated into other languages more easily), portability (e.g. developing an iPhone application to be compatible with other smart phone platforms like Android or Windows Phone) accessibility (e.g. the program complying with ADA guidelines), availability (e.g. the system will be able to reboot in under 2 minutes in the event of failure.), etc.

**Tasks** consists of actual product development steps for each user story. The **ID** column specifies the task for a user story ID. The **Resource** column specifies the name of team member associated with a corresponding task ID >>

**Sample User Stories:** *This sample assumes SPRINT 2 is completed. For SPRINT 3, user stories are already planned along with* ***Acceptance Criteria*** *and* ***Verification Description****.* ***Tasks*** *will be added when SPRINT 3 is completed.*

**SPRINT 1**

**Estimated User Story Points:** 2

**Actual Completed User Story Points:** 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| 100 | Onset | **As a** conference attendee,  **I want** to be able to register online,  **So that** I can register quickly and cut down on paperwork | **C** | **2** | **3** | | **100%** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| **110** | A user cannot submit a form without completing all the mandatory fields | | **Create a test case to verify non-empty fields.** | | | | |
| **111** | Information from the form shall be stored in the registration database after form submission | | **Create test case to verify information is stored in the database.** | | | | |
| **112** | Payment shall be accepted via credit card | | **Create test case to verify credit card payment method from bank.** | | | | |
| **113** | An acknowledgment email shall be sent to the user after submitting the form. | | **Create test cases to verify sending of acknowledgement email after successful payment.** | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| 1 | Create a registration page with all required fields (FName, LName, organization, Address details, email, credit card details) and register button at the bottom. | | | | | **Team member 1** | |
| 2 | Develop a backend functionality that checks required fields are non-empty when user clicks on register button. | | | | | **Team member 2** | |
| 3 | Built a functionality which verifies payment from bank based on credit card details at 1. | | | | | **Team member 2** | |
| 4 | Built a functionality which stores registration data in the database according to the specifications and sends acknowledgement email to the registered email else display payment failure message. | | | | | **Team member 3** | |

**SPRINT 2**

**Estimated User Story Points:** 8

**Actual Completed User Story Points:** 8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | | **Actual Equivalent Story Points** | **% Completed** |
| 200 | Onset | **As a** customer,  **I want** to withdraw cash from an ATM  **So that** I don't have to wait in line at the bank. | **C** | **8** | | **8** | **100%** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| **210** | ATM card shall be valid to be accepted | | **Create test cases to verify ATM card details with the banking system.** | | | | |
| **211** | ATM shall only try to dispense cash if it has enough to cover the amount requested | | **Create a test case to check cash in ATM dispenser.** | | | | |
| **212** | When customer requests cash, saystem shall ensure that amount withdrawn <= accountBalance | | **Usability test to determine if customer can enter required cash in the system.**  **Create test cases to check entry of valid amount.** | | | | |
| **213** | System shall ensure that account is debited, cash is dispensed and card is returned. | | **Create test cases to verify the following steps for a transaction: account data is modified, dispenser dispenses the cash, and card is returned back.** | | | | |
| **ID** | **Tasks** | | | | **Resource** | | |
| 1 | Develop a method which reads card details (card #, account #, validity) from the ATM card in the machine and verifies details from the banking system. If details are not verified, card rejection message shall be displayed and the card shall be returned to the customer. | | | | **Team member 1** | | |
| 2 | The developed function shall also include a verification technique which confirms that the ATM machine contains enough cash for the withdrawal. | | | | **Team member 2** | | |
| 3 | Create cash withdrawal page with submit button for customers where they can enter the required amount for withdrawal and press the submit button. | | | | **Team member 2** | | |
| 4 | Build a method which shall verify that the amount entered <= the amount available in the account else displays rejection without any transaction and returns card to the customer. | | | | **Team member 3** | | |
| 5 | Create a function which shall complete the transaction successfully in the following steps:   1. accountBal = accountBal – amountWithdrawn 2. Cash shall be dispensed by the ATM machine.   Card shall be returned to the customer | | | | **Team member 3** | | |

**SPRINT 3**

**Estimated User Story Points:** 55

**Actual Completed User Story Points:** N/A

**Main User Story:** “As the HR manager, I want to create a screening quiz so that I can understand whether I want to send possible recruits to the functional manager.”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | **% Completed** |
| 300 | Onset | **As a** manager,  **I want** to browse my existing quizzes  **So that** I can recall what I have in place and figure out if I can just reuse or update an existing quiz for the position I need now. | **C** | **8** |  |  |
| **Acceptance Criteria** | | | **Verification** | | | |
| **310** | Manager shall be able to search by quiz name. | | **Create test cases to verify search results by quiz name.** | | | |
| **311** | Manager shall be able to search by quiz topics included. | | **Create test cases to verify search results by quiz topics.** | | | |
| **312** | Manager shall be able to search by creation and last used date. | | **Create test cases to verify search results by creation and last used date.** | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | **% Completed** |
| 400 | Onset | **As an** HR manager,  **I want** to match an open position’s required skills with quiz topics  **So that** I can create a quiz relevant for candidate screening. | **C** | **13** |  |  |
| **Acceptance Criteria** | | | **Verification** | | | |
| **410** | Manager shall be able to search the quiz topics by name. | | **Create test cases to verify search quiz topics by name.** | | | |
| **411** | Quiz topics shall be allowed to have alternate names, terms for searching | | **Create test cases to verify alternate quiz names and terms to search.** | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | **% Completed** |
| 500 | Onset | **As an** HR manager,  **I want** to send a draft quiz to the functional manager  **So that** I make sure I’ve covered the right topics on the screening quiz. | **C** | **13** |  |  |
| **Acceptance Criteria** | | | **Verification** | | | |
| **510** | Manager shall be able to add another user by email in this flow. | | **Think about verification criteria.** | | | |
| **511** | Manage shall be able to include notes and customize the email. | |  | | | |
| **512** | Manager shall be able to copy a link (for case where HR manager wants to send their own email). | |  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | **% Completed** |
| 600 | Onset | **As a** functional manager,  **I want** to send feedback on the screening quiz to the HR manager  **So that** I make sure I’m getting the best possible screening on candidates. | **C** | **21** |  |  |
| **Acceptance Criteria** | | | **Verification** | | | |
| **610** | Manager shall be able to supply comments in-app. | | **Think about verification criteria.** | | | |
| **611** | The comments above shall be quiz-specific by default but can also be general. | |  | | | |
| **612** | Manager shall be able to easily copy the name or URL of the quiz for their own correspondence | |  | | | |

## 

## 4.5 Constraints and Limitations

<< This section provides a list of constraints and limitations for the project. This provides additional information about any limitations that may exist in the project (e.g. will not work in versions of Internet Explorer prior to 8.0) that are not covered by requirements, but provide important information related to the project. This may include constraints related to security, performance, and other aspects of the system.

The **Constraint** field lists the constraint or limitation for the project. The **ID** field lists the related requirement ID (if any) that provides additional context for the constraint. >>

**Example:**

|  |  |
| --- | --- |
| **Constraint** | **ID** |
| Team will not provide alternative language pack files. | 720 |
| Provided 3D models will not be of professional-grade quality. | 1100 |
| Team will use Visual Studio 2007 for development as 2010 is not available. |  |

# 5. Design

## 5.1 Introduction

<< Briefly describe the major aspects of the design and, if applicable, how a developer will use it. **Example:** “Create and post a General Ledger transaction using the glTrx routines. Perform account inquiries with gjJournal routines.” >>

## 5.2 Scope

<< Provide a brief overview of the scope of this design. Also touch on anything that will not be included in this document.

**Example:** This document is a high level overview of the design decisions and paradigms. The primary component that needs design is the R visualization library which has a variety of functions for creating different visualizations. An example of Azure ML machine learning will be shown using the cloud based tools, this is primarily designed through a UI and supplemented by custom R scripts that will be written as needed. The MapReduce function is likely beyond the scope of this project, and thus will not be included in the initial revision of this design document. >>

## 5.3 High-Level Component Design

<< Create a diagram of the high-level components or modules in the program, linking them with arrows to show any dependencies. Also complete the tables to provide a description of each module as well as the table which traces components to their related requirements. >>

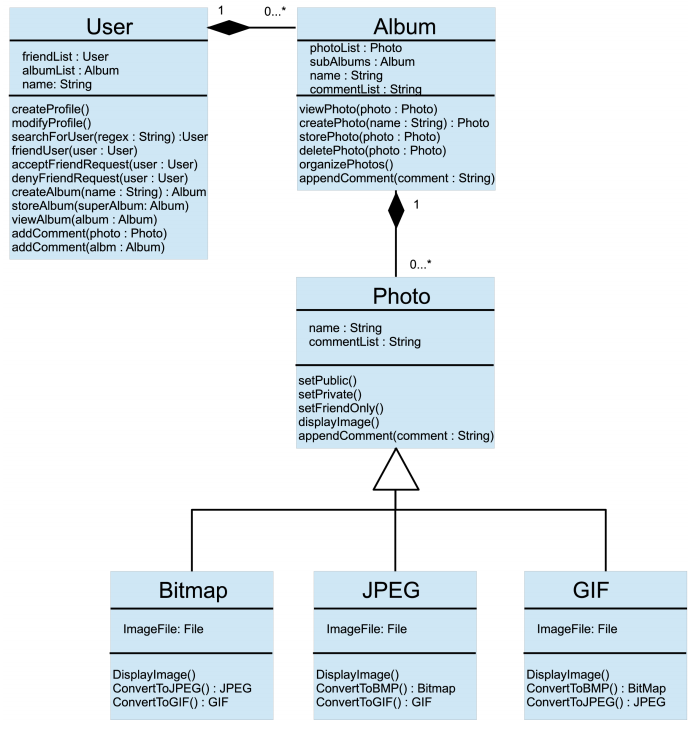
**Example:**

|  |  |  |
| --- | --- | --- |
| **Component** | **Related Requirements** | **Description** |
| R Data Import Function | 100, 300 | Takes the reduced telemetry data, imports to R, cleans and orders the correct fields |
| R Statistics Function | 300 | Allows the user to choose from a variety of statistics about an individual session or form, or an overview of the entire data set. |
| R Visualization Function | 200, 900 | Allows the user to choose the type of visualization desired and generate any of number of them based upon the data that was imported |
| Machine Learning Model | 400, 500, 1000 | Takes data from a test data set and ultimately predicts the users next action, this will ultimately be able to dynamically process telemetry as it happens |

## 5.4 Class Diagram

<< Include a class diagram for all classes to be designed. Optionally include major data elements of those classes and important methods and functions that will be used by other classes. >>

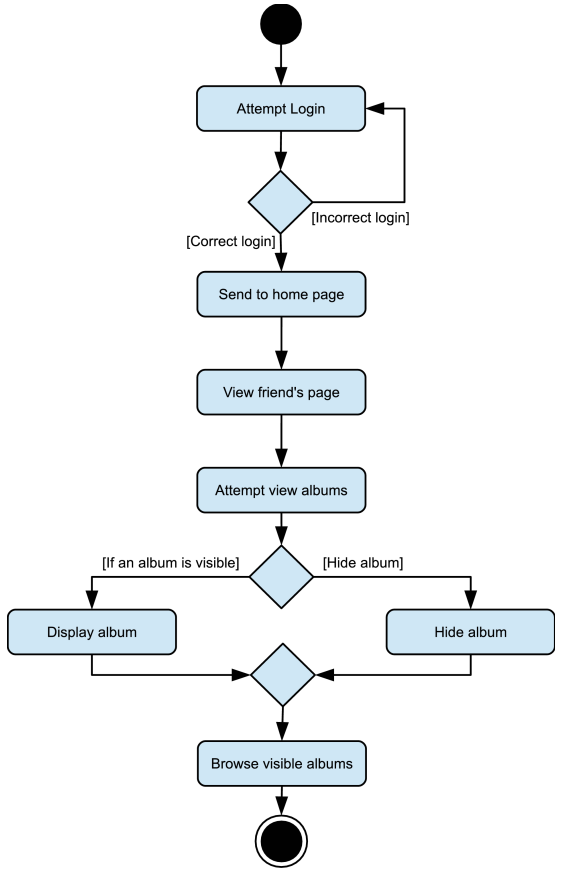
**Example:** Online Photo Collection



## 5.5 Activity Diagrams

<< Include activity diagrams for important workflows in the program. At least one diagram should be included for the main workflow in the program. Optionally include labels that indicate which component is responsible for that part of the activity. Activity diagrams for components which perform complex tasks should also be included. >>

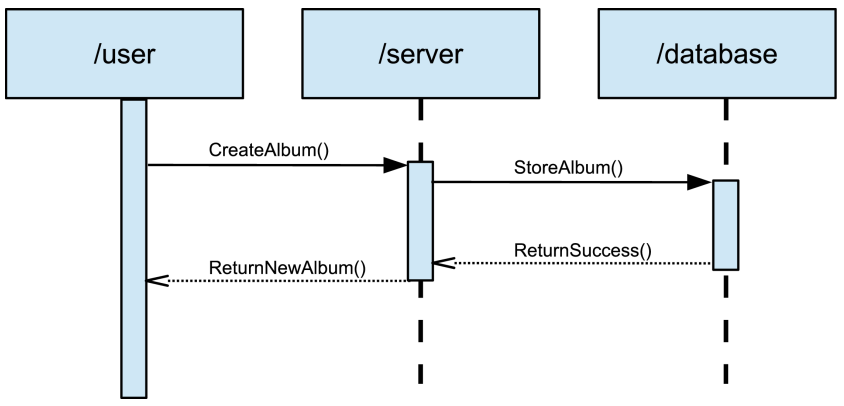
**Example:** View friend’s photo album



## 5.6 Sequence Diagram

<< Include sequence diagrams for important functionality of the program to indicate control flow. These diagrams should include classes found in the class diagram and use the methods for those classes to show the interaction between them. >>

**Example:** Create new album



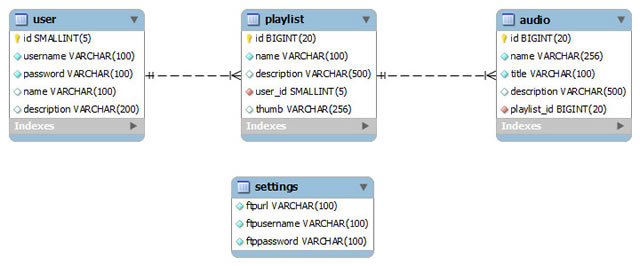
## 5.6 Alternative Designs and Design Rationale

<< Provide information regarding other designs which were considered, but not chosen. Provide rationale for why the chosen design was selected over the alternative candidate designs. >>

## 5.7 Data Architecture

<< Include any information or diagrams that provide details about databases, xml configuration files, or other data structures that are a part of the system. If a very specific format is required, it may be worthwhile to provide a more robust description or a detailed design such as a database schema. >>

Example: Online Audio Gallery



# 6. User Interface

## 6.1 UI Description

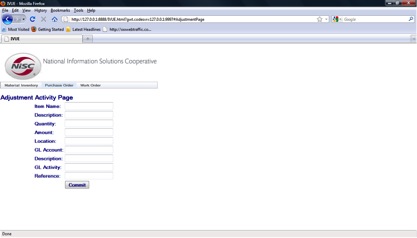
<< Provide a brief description of the UI that will be used in this program and how users will interact with the program.

**Example:** We are creating a research project using R and Azure ML. R is a statistical programming language that uses the R console, which will be the primary means of interacting with our code. This uses standard console UI, and it is not in the scope of our project to create a UI on top of that. The Azure ML machine learning model has its own built-in drag and drop UI for the different code modules we create, so there is no need to generate any sort of AI for that portion of the project either. >>

## 6.2 UI Mockup

<< Create a mockup of the user interface. This can be a simple drawing that demonstrates key parts of the user interface or a screenshot of a prototype created within an IDE. >>

**Example:**

****

# 7. Project Closure

## 7.1 Goals / Vision

**<<** Provide an update to the vision statement that was originally stated in the Project Initiation document.

**Example:** Our original goals for this project were to take telemetry data from Dynamics AX, analyze and visualize that data, then create machine learning models in Azure ML to predict user navigation based upon previous actions. Through the course of the project, these goals were altered so that the primary goal became creating a well-documented, extensible R language package that facilitated cleaning and importing telemetry data, and contained a variety of useful analysis and visualization functions to make the raw data more understandable. >>

## 7.2 Delivered Solution

<< Provide a high-level description of what was planned and what is being delivered.

**Example:** Our solution delivered primarily consisted of a fully documented, fully featured R language package that contained the functions for importing telemetry data, cleaning, separating, and isolating that data, then performing a variety of analysis, statistics, and visualizations on that data. This R package has standard R documentation, a full suite of unit tests, and an integrated manual and help documentation to allow anyone with a basic familiarity with the R environment to utilize our functions. >>

## 7.3 Remaining Work

<< Provide a short summary of what should be done next, ways of further improving the project, or any additional recommendations.

**Example:** We created a prototype of a web-based UI for our R package using Shiny in an attempt to make the functionality more accessible for those unfamiliar with R. There are a few minor functions implemented, and a basic UI setup, but a good deal of work would be required to flesh out this web app with all of the functions currently available in the R package, should Microsoft choose to pursue this UI overlay further. >>

# 8. Definitions and Acronyms

<< This section provides a definition for terms or acronyms used in this document which may not be familiar for all users. >>

**Example:**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Enterprise Resource Planning (ERP) Software | Software that tracks payroll, inventory, labor, capital, etc. for businesses to increase efficiency and manageability. |
| Dynamics AX | Microsoft’s popular ERP software solution with the capabilities to manage small and large businesses. |
| Microsoft Azure | Microsoft’s cloud computing platform that houses modules such as Azure ML (Machine Learning) and Azure HDInsights (Hadoop). |
| R | Open source statistical analysis software used by many developers in data science and analysis. |
| Form | A window or application page in Dynamics AX that contains a set of controls or functions. |