AMM

# Turk System
# Software Requirements Specification For Web App

**Version 1.0**

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 13/10/2017 | 1.0 | First version of web based Turk System | Matthew Jacome<br>Mohamed Khelif<br>Abraham Villaroman |
| | | | |
| | | | |
| | | | |

# Table of Contents

# Software Requirements Specification

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to present a detailed description of our web based Turk System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which must operate and how the system will react to external stimuli. This document will also cover hardware, software, and various other technical dependencies.

### 1.2 Scope

Our web based Turk system provides an environment for registered clients to post requests for a system they want created. Registered developers are able to see these system requests and can post an asking price for creating the requested system. The client who made the request is able to hire one of these developers. Any issues between clients and developers as well as all transactions are handled by a super-user. Information related to clients and developers involvement in past systems are made public as well as a typical rating system. Non-registered users are able to view general information made public by clients and developers but cannot take any action without registering first.

### 1.3 Definitions, Acronyms, and Abbreviations

| Terms | Description |
|---|---|
| Registered User | Those that are registered on the web app. |
| Super-user (SU) | Registered users who handle other users accounts, money related issues, and proctor user activities. |
| Client | Registered users who post system requests. |
| Developer (DEV) | Registered users who implement systems requests post by clients. |
| Visitor | Non-registered users who can only see information made public by registered users. |
| JS (Java Script) | Scripting language commonly used on websites |
| NodeJS | A javascript Run time based on the Google V8 Javascript Platform |
| React JS | A javascript library for creating user Interfaces |
| Database | Collection of all the information monitored by this system |
| Node-Server( Node Js Server) | Server running Node js Program |
| HTML | Hypertext Markup Language, a standardized system for tagging<br>text files to achieve font, color, graphic, and |

| | |
|---|---|
| | hyperlink effects on<br>World Wide Web pages. |

### 1.4 References

"React." A JavaScript Library for Building User Interfaces, Facebook Inc, reactjs.org/.

"node js", a JavaScript run-time environment for executing JavaScript code server-side. , nodejs.org/en/.

"Styled-Components". A JavaScript library that allows the us to style our User Interfaces with ES6 and CSS syntax. https://www.styled-components.com/

"Express" is a Web Framework that makes creating Web Apps and API 's , https://expressjs.com/

"Mongodb" is a noSql database items are stored in JSON and Querys, https://www.mongodb.com/

### 1.5 Overview

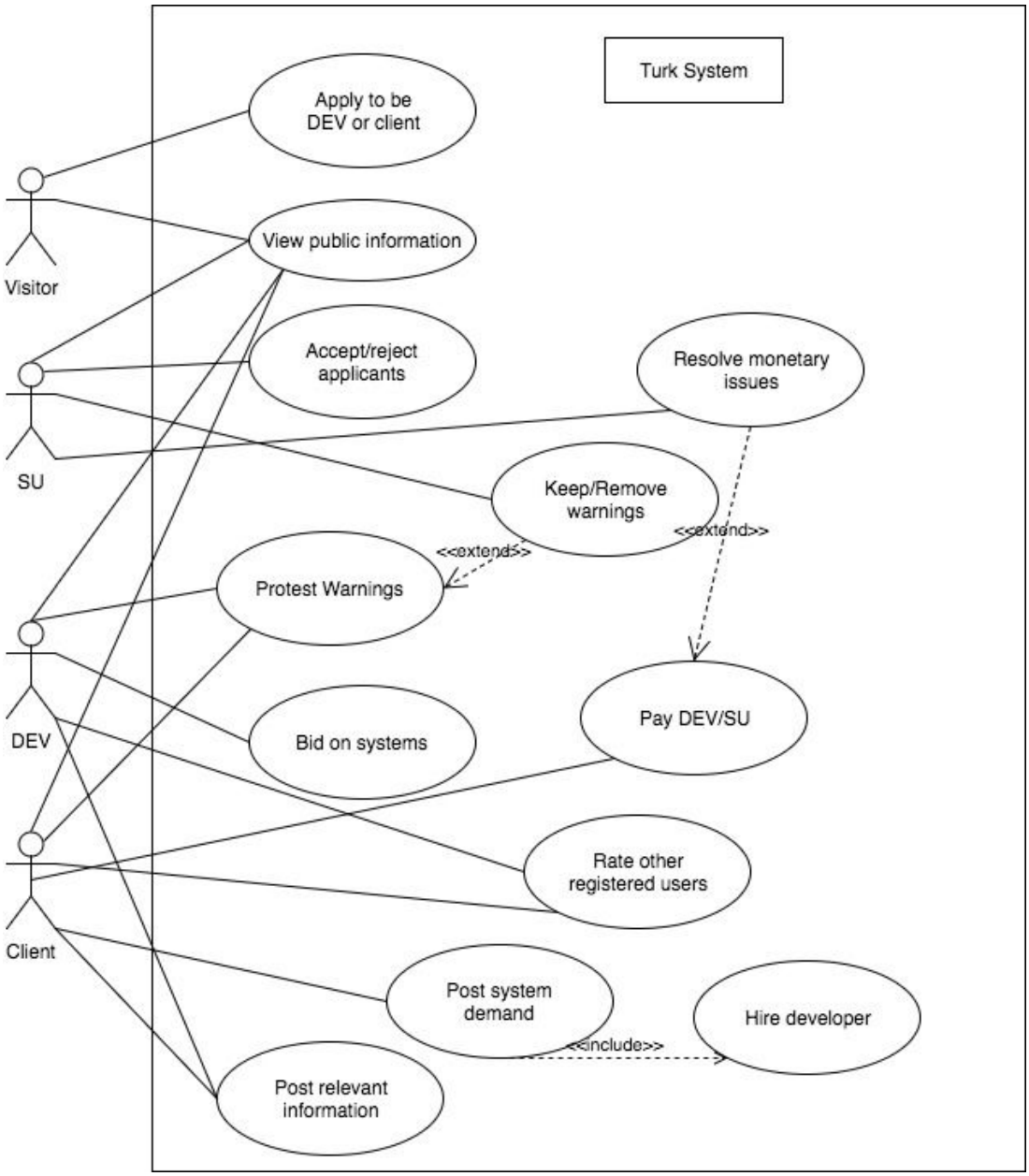
The Overall Description section of this document will give an overview of the functionality of the product. It will describe the informal requirements and will be used provide context for the technical requirements in the following sections.

The Specific Requirements section is mainly used for developers and describes in technical terms how each function of the product should work.

## 2. Overall Description

### 2.1 Use-Case Model Survey

The use case diagram below helps to visualize the functionality of our web based Turk System and helps to show how each users interacts with the system.




There are four users in our system: visitors, super-users (SU), developers (DEV), and clients.

Visitors are users who are not registered or not logged in. They can only see public information and apply to be either a developer or a client.

Developers are registered users that are logged in. They can view public information, bid on systems posted, rate other registered users, protest warnings given, and post relevant personal information.

Clients are registered users that are logged in. They can view public information, post system demands, pay developers and SUs, protest warnings given, and post relevant personal information.

Super-users are registered users that are logged in. They can view public information, accept or reject applicants, resolve monetary issues, and handle warnings given to other registered users.

## 2.2 Assumptions and Dependencies

It's assumed that there will always be at least one super-user who is active at all times to perform his duties.

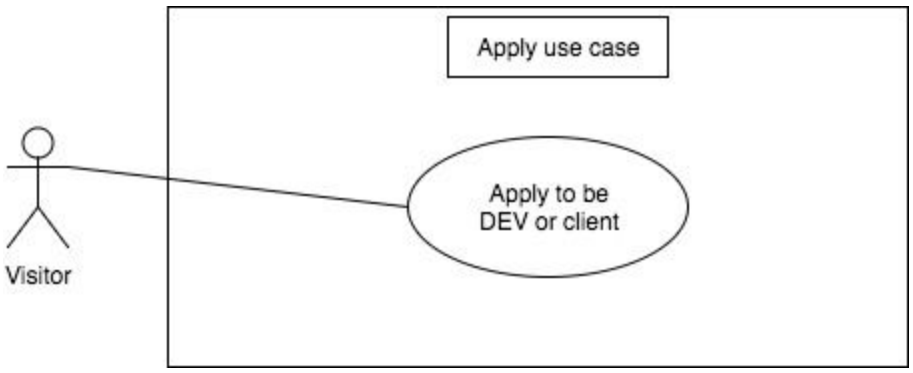
Login credentials, user information, past projects and other necessary information should be stored somewhere such as a server in a way so that it can grow and accept new information at any time.

# 3. Specific Requirements

## 3.1 Use-Case Reports

Use case: Apply for registered user

Diagram:

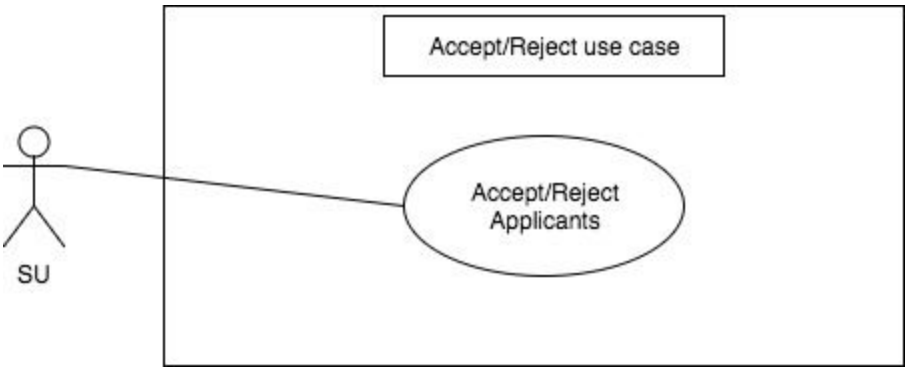



Brief description: A visitor on our web app can apply to be either a developer or client

1. The visitor must choose to be either a developer or a client
2. Basic information is given from the visitor
3. The visitor must deposit a set amount of money
4. A unique username and password must be assigned

Use case: Accept or reject visitor application

Diagram:

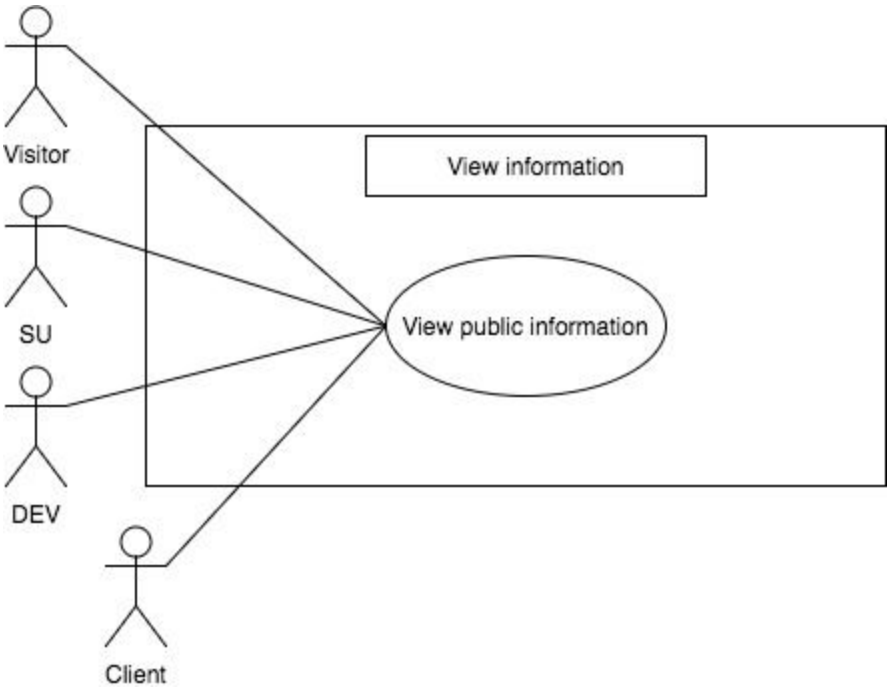



Brief description: Only a super-user who is logged in on our web app can accept or reject visitor applications

1. Super-user must first check the credentials and information given from the visitor applying
2. For any rejection, super-user must give a one sentence reason
3. Visitors in a black list are automatically rejected
4. Accepted visitors are asked to give more personal information

Use case: View public information

Diagram:

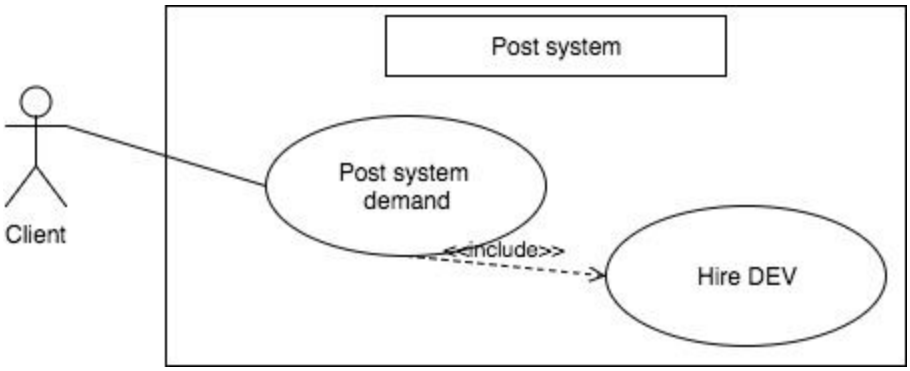



Brief description: Any user can view information made public

1. Any user who is on our web app can see public information
2. This information is posted and made public by registered developers and clients

Use case: Post system demand

Diagram:

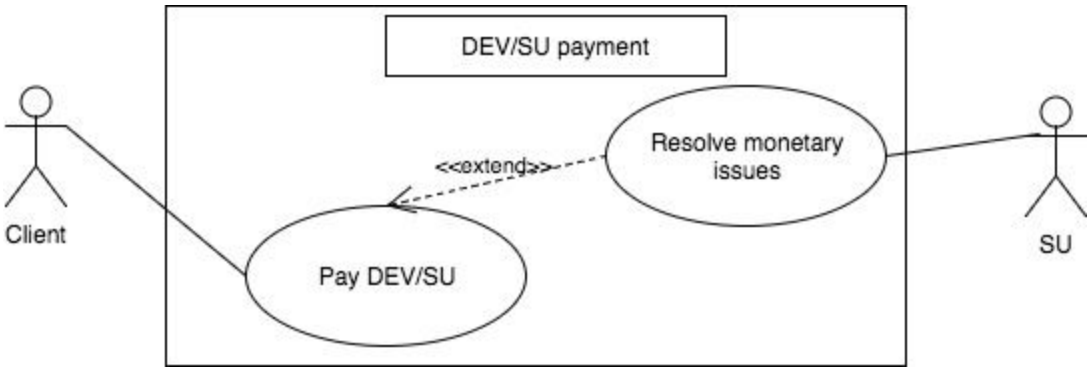



Brief description: Registered clients can post systems they want built. From there they hire a developer to implement it.

1. A client can post a system demand along with a paragraph describing the system specifications and a bidding timeline is given.
2. The client chooses to hire a developer that has place a bid on their system.
3. If the client doesn't choose the lowest bidder, an explanation is given for the choice they made.

Use case: Developer/Super-user payment

Diagram:

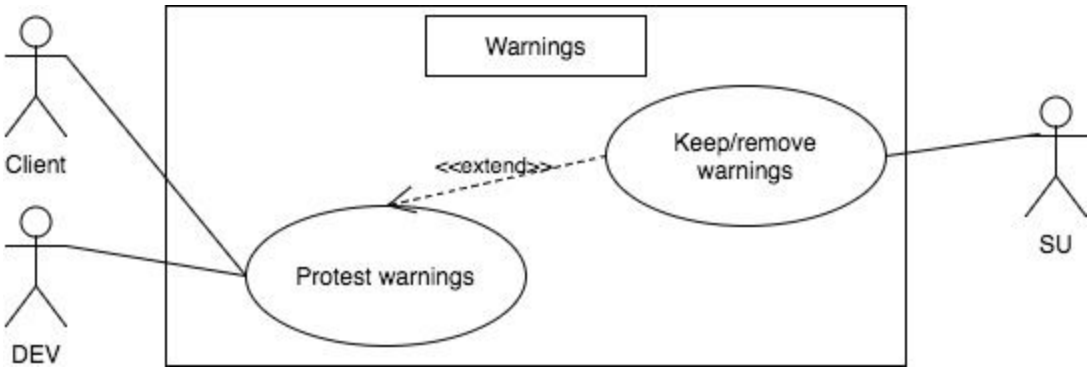



Brief description: The client must give payments to the developer working on his/her system as well as the super-user at different stages of the system's development.

1. Half of the bidding price must be given to the chosen developer once chosen.
2. If the developer delivers the system after the deadline, a fixed penalty from the first half is given back to the client and the developer is given a rating of 1.
3. If the developer delivers the system before or on the deadline, the second half of the bidding price is given to the developer.
4. The client is also asked to rate the delivered system. If the rating is less than 3, the developer must give a note describing the reasoning for the rating. The super-user must then step in and decide payments after communicating with both client and developer.
5. For every transaction, 5% of the bidding price is paid to the super-user from the client and developers account.

Use case: Protest warnings

Diagram:



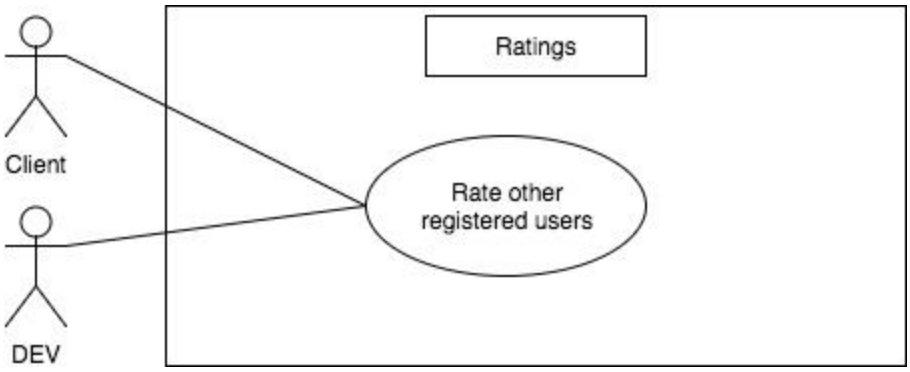

Brief description: Warnings are given to clients and developers for suspicious activity are given warnings which they can protest to their super-user.

1. Clients and developers who have an average rating of less than or equal to 2 for 5 or more systems are given a warning for poor performance.
2. Clients and developers who on average give ratings of less than 2 or greater than 4 for 8 or more systems are given a warning for irresponsible evaluations.
3. Clients and developers can protest the warnings they've been given.
4. The super-user can choose to remove the warning or keep the warning on the client or user.

5. Registered users who have more than 2 warnings are removed from the system and put on the blacklist for 1 year.

Use case: Rate other users

Diagram:

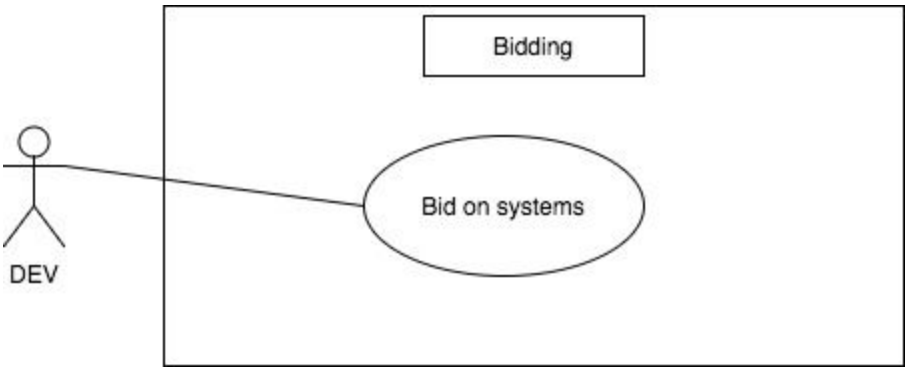



Brief description: Clients and developers are able to give others clients and developers ratings.

1. The client gives the developer a rating based on the system delivered. An explanation is given for rating less than 3.
2. The developer can also rate the client after transactions are complete. An explanation is given for a rating less than 3.
3. Warnings are given based on ratings.
4. Ratings are also made public.

Use case: Bid on systems

Diagram:

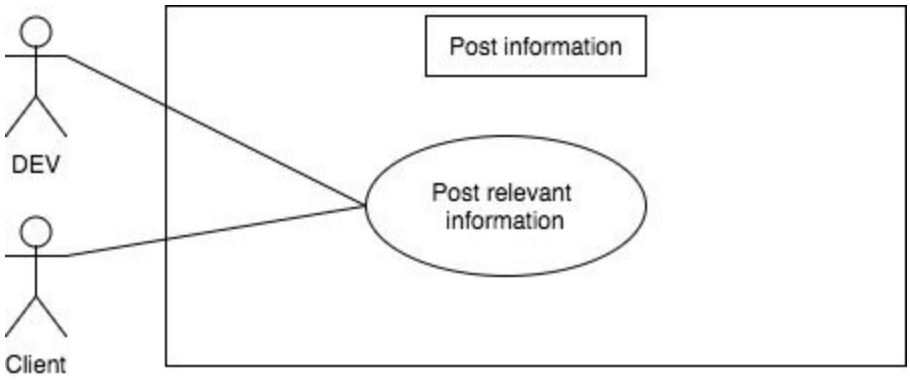



Brief description: Developers are able to bid on system.

1. Registered developers who logged in are able to to place bids on systems posted by registered clients.
2. A time frame to complete the system is given.

Use case: Post information

Diagram:




Brief description: Registered logged in users can post relevant information.

1. Registered users are asked to put in more infomation such as a resume, picture, and interests.
2. Developers are asked to give some sample work.
3. Clients are asked to give their business credentials.
4. The history of clients and developers, including ratings and project details, are made public.

### 3.2 Supplementary Requirements

For the Turk System to remain stable and efficient, the following requirements should be met.

1. Multiple super-users at all times

   As the system grows, more conflicts are likely to occur. The number of super-users should increase to ensure there is at least one active at all times. There needs to some selection process to ensure these super-users are unbiased and show fair judgement when resolving conflicts.

2. Up to date information

   The information on the website should be maintained up to date, such as systems that can be bid on and users personal information, in order to prevent any overlap that might occur.

3. Maintenance and changes on websites user interface

   The website interface needs to be kept maintained to handle the possible large amount of information presented as the system grow. It should also be kept user friendly for all registered users and any visitor that finds the system.

## 4. Supporting Information

[The supporting information makes the **Software Requirements Specification** easier to use. It includes:

- Table of Contents