KC CAB DRIVER APPLICAITON

Abstract

A driver-interaction application for the KC Cab company to allow drivers to coordinate rides with customers via a mobile app

Josh Basic
Jacob Davis
David Gamino
Paul Horner
Erin Parsons

Contents

Introduction

Initial Description

The person requesting this application is Stephen Jennings the owner of KC Cab LLC in Ellensburg WA and he wants a mobile application for his company. His request is for two separate applications, one for the person requiring a ride and another app for the drivers; it should provide a simple free service that allows users to easily request to be picked up.

Purpose

KC Cab hopes to increase service availability by providing a free-to-use app to request rides within the Kittitas area. This app would also interface directly with drivers to coordinate rides.

Scope

- Collection of fares is outside of the scope of this project and will be handled in-cab.
- Due to the scale of this project, two teams will work for the client; one team to handle driver interaction, and one team to handle client interaction. This document applies to the driver interaction component.
- The driver-interaction component will run on a standard Android platform.
- The developed app will operate within the Kittitas area and will not accept ride requests from outside the Kittitas area.
- Design and development-operation of the operational database and server will be handled by the driver-interaction team.

Definitions

Term	Definition
Client	Refers specifically to the KC Cab Company
Driver	A cab driver working for KC Cab
DBMS	Database management system

Product Description

Product Perspective

The developed application will be used by cab drivers to accept ride requests and view pick-up locations. One of the primary requirements is to limit interface complexity, as the application is intended to be used by the operator of a cab. The developed application will differ from traditional rider-request applications in that driver interaction will be limited to accepting ride requests, viewing pickup locations, and basic ride tracking information. These interactions will be available from the main interface, which will consist of a map, a pickup queue, and a limited collection of tools such as session management (sign-in/sign-out) and light/dark theme selection.

The provided interface will be backed by a connection to a remote server which will provide ride requests, pickup locations, and other information pertaining to accepted rides. The remote server will run a database containing live ride requests and historical information and will provide authentication for drivers. Rides can only be accepted by signed-in drivers, and all rides and events will be linked to the relevant driver for administrative oversight and analysis purposes.

- The developed application will communicate with the remote sever using a cellular network.
- Map provider: TBD (currently planning on Google Maps)
- Application platform: TBD (currently planning on Xamarin)
- DBMS: TBD (currently planning on SQL2017)

Product functions

- Upon launch, the developed application will authenticate the driver before allowing any further action.
- Upon authentication, the driver may view the queue of rides to accept, and may accept rides at their discretion, with some limitations imposed by the client.
- Upon accepting a ride, rider information and pickup location will be made available.

Additional controls become available relating to active rides:

- Cancel ride
- Ride picked up/completed

User Characteristics

The target user of the developed app has no expertise in device operation besides ensuring access to required data (GPS & Cellular service). The target user may attempt to interact with the application while driving, necessitating a simple interface, large text, and other visual cues to reduce attention requirements on the user.

Specific Requirements

1. Security:

    1.1. Driver accounts will be protected with a username and password.

    1.2. Driver account data will be secured on the remote server.

    1.3. Rider information will be secured on the remote server.

    1.4. Communication between the application and remote server will be properly secured with SSL, TLS, or another appropriate system.

2. Available Rides:

    2.1. A driver can accept the oldest ride from the queue.

    2.2. A driver can mark the pickup time of a ride after accepting a ride.

    2.3. A driver can mark a ride as complete after marking the pickup time.

    2.4. A driver can cancel a ride after accepting it but before marking the ride as complete

        2.4.1 A cancelled ride returns to the queue.

        2.4.2 A note of cancellation is made available to the client app (displaying this notification is outside of the scope of this project).

    2.5. Pickup location is provided to the driver for the current ride

2.5.1 Pickup location is displayed on the interface's map.

2.6. One-click to contact rider is provided for accepted rides

2.6.1 TBD

Potential features

1. Administration page

1.1. Query rides by time

1.2. Query rides by driver

1.3. View live queue

1.4. View statistics by driver

1.5. View global statistics such as average wait time

1.6. Administer accounts

2. Provide client with live map of driver position.

3. iOS support.

Non-Functional Requirements

1. All cabs must run the app simultaneously.

1.1 Nine cabs exist, and all must be able to run.

2. The delay between notification must be less than one minute.

Design Constraints

Application must be simplistic and easy to use.

Priority

Development priority will largely focus on the high-functionality features initially, as well as security. This will allow for quick demonstration, enabling the client to provide feedback on the application as it progresses, and potentially direct future development moves if changes are desirable. Additionally, by including security as an initial priority, the application design phase can avoid some of the more common security failings such as using unencrypted communications by default. After core functionality is achieved, development will continue in standard requirements and potential features as directed by the client.

Analysis

Feasibility Analysis

As of this writing, the client has not made any infeasible requests, either hardware or development-time wise. Completion of the required specifications for the driver-interface component is projected well within the time limit provided, likely with time to address some of the apportioned requirements before the end of the Winter quarter. The largest potential difficulties will be working with the DBMS, which our team is largely unfamiliar with, as well as determining the specific protocols to interact with the client-interface component, as it is being developed by another group.

Consistency Check

As of now, all requirements work toward a functional application and do not conflict.

Validity Check

All features requested for this application work toward the original task for this application. No features are outside the scope of the design.

Database Design

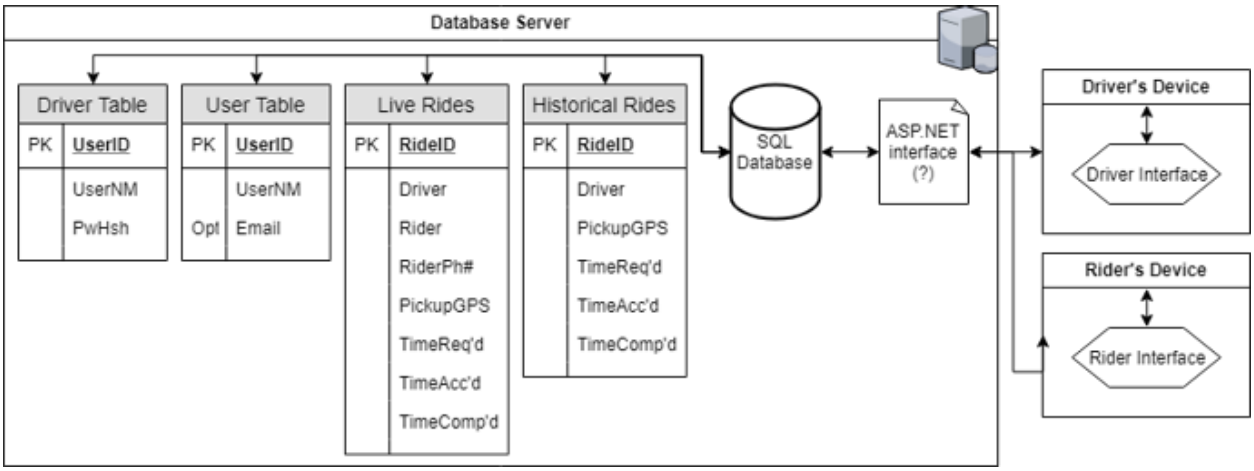

Figure 1. A visual description of the SQL Database and its connectivity.

The database will consist of four tables driver, user, live rides and historical rides. These tables will be shared between the Driver's application and the Rider's application.

Drivers Table

Will store the drivers accounts. It will only consist of user numbers and a hashed password. The key will be a user ID.

User Table

This table stores the user number and email which is optional, so it may be null. The key will be user ID.

Live Rides

This is a more substantial table which will be used by both applications during the time the ride is still active. The table holds all relevant data its columns are the driver, rider, rider phone number, pickup GPS location, time requested, time accepted and time completed. The key is the ride ID.

Historical Rides

Will store the rides that have already been completed. It will maintain only some of the data from the live rides including driver name, pickup GPS location, Time requested, time accepted and time completed.

High Level Design

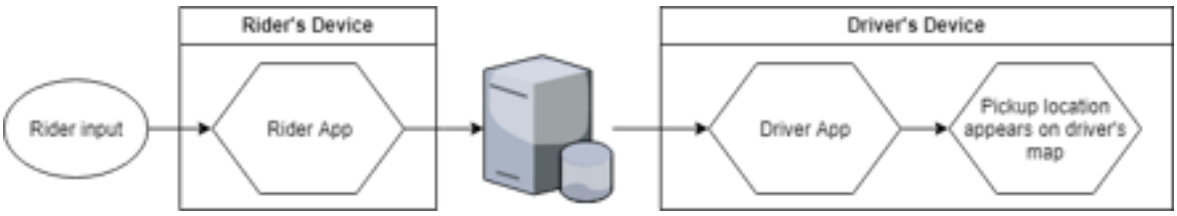

Figure 2. A high-level general design of the base function of the application and its communication with the server and rider application.

Rider Application

The input provide by the rider will be collected by this application (created by the other team) and sent to the server to be stored in the database.

Server Database

The database will take the user input and store it in the appropriate Live Ride table for the use of the driver application.

Drivers Application

The driver application will require a sign in from the user that will then be checked using the database. Once the account is verified the application will display the contents of the Live Ride table on the main data display page of the application. Once the driver selects a rider the interface will bring them to a new page with the map.

Map Display

This page will contain all the rider's data including the name, phone number and location which will be displayed on a map. This will also contain a finish ride button which once selected will bring the driver back to the main data display page

Interface Design

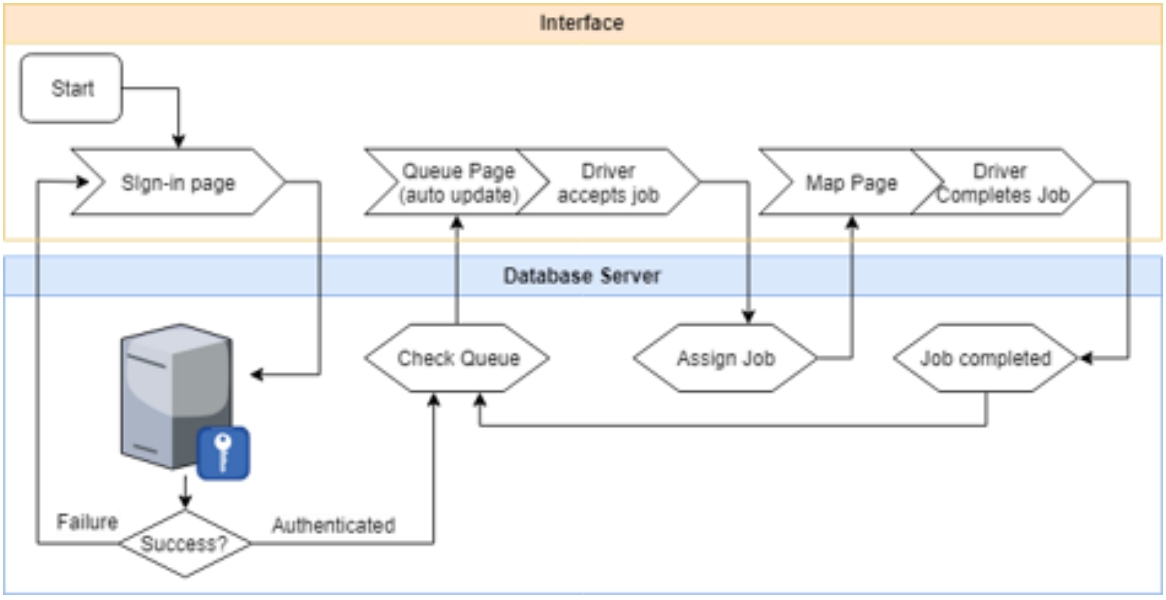

Figure 3. Interface Design demonstrating the interaction between the user interface and database.

This diagram represents the process the application goes through when it is first opened; It shows the interactions between the user interface and the database server. When the application is first opened, it presents a sign in page and once a user inputs a username and password, it then is checked against the database. If the information is invalid it will request valid credentials; if the input is authenticated, the ASP Net interface will check the live ride queue data and present it on the requested rides main screen. Once the driver accepts a job, the interface will assign the job to that driver and present the map page on screen. When the driver completes the job, the ride will then be moved from the live rides table to the historical rides table and the interface will once again check the live ride queue.

Schedule

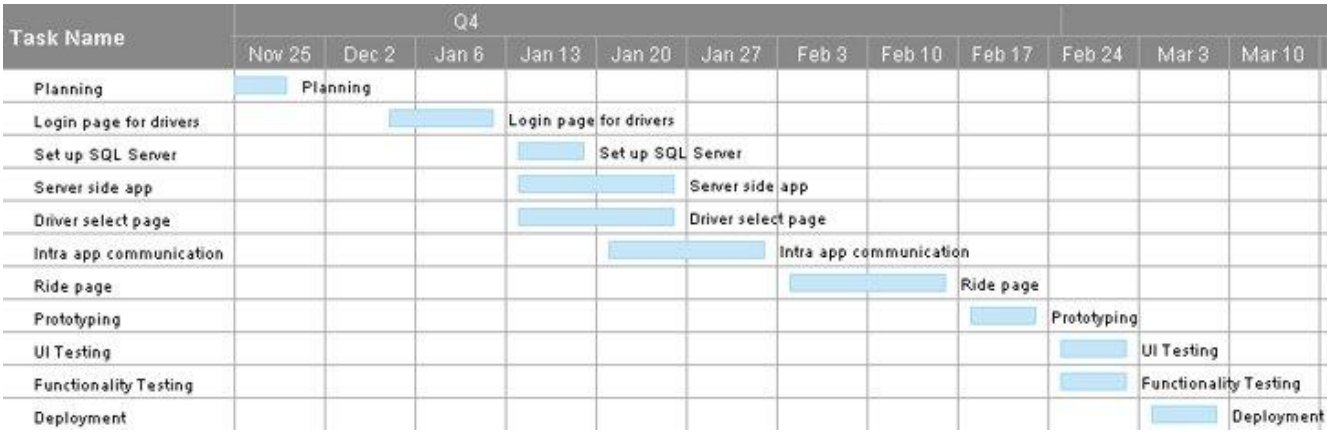

Figure 4 Gantt Chart of the software development process broken down by week and process.

The chart is a planned development schedule for the projects remaining time. We have just recently finished the planning stage of the development process and will be starting the development stage the first week of December. As displayed above we will work on the login page through all of December completing it the first week of January. The second week of January will start the development of the SQL Server along with the Server-side application and Driver select page the goal is to complete these sections by the end of the third week of January. The second week of January will start the Intra application communication which involves connecting the driver's side application to the rider side application using the database the plan is to complete this by January. The start of February will involve the development of the ride page for two weeks and once this is completed the prototyping stage will begin. Prototyping, UI Testing, Functionality Testing and Deployment will be the only things done for the last four weeks. Testing will be done throughout the process as different tasks are completed.