Tiger Tickets

Alex Lam (aklam@princeton.edu)
Evelyn Karis (ekaris@princeton.edu)
Jason Shi (guoweis@princeton.edu)
Richard Du (rdu@princeton.edu)
Henry Birge-Lee (birgelee@princeton.edu)

Section 1 ("Overview")

Tiger Tickets will be a web-app that provides a centralized location for student and faculty to buy and sell tickets for on-campus shows at the original price offered by the Frist Ticketing Office. Money transactions will not be handled by our application, rather, it will simply bring people together to trade, buy, or sell tickets to events that are sold out. Our web-app will acquire the relevant information for each outstanding show from the Frist Ticketing Office event page. We hope that this will provide a more efficient market for tickets to sold-out student shows and alleviate the current burden on res-college list servs to redistribute tickets

Section 2 ("Requirements and Target Audiences")

Tiger Tickets will solve the problem of unsold/unused tickets for Princeton student events leading up to the date of the event. The existing situation is highly decentralized and forces students who cannot use their ticket(s) for a show or are looking to buy tickets for a sold out show to spam various listservs (such as ButlerBuzz, WhitmanWire, Forbes Reinnformer, various student-group listservs etc.) to get rid of their ticket or purchase a surplus one. Because each student only has access to one college listserv, they can only access a small proportion of the student population when looking to get rid of, buy, or trade a ticket. Because this decentralized model is highly inefficient, many students end up not being able to sell or exchange their tickets. In order to remedy this situation, we wish to centralize this process, so that all students who want tickets or want to get rid of tickets are on the same market.

Section 3 ("Functionality")

We are designing our platform to facilitate ticket transactions between students. Consequently there will be three major parties: ticket buyers, ticket sellers, and ticket exchangers.

Ticket buyers will use our service if the Frist ticket office has sold out all of its tickets for a particular show or show night. Optional functionality: have people be able to put multiple days/times as options for the show they want to see.

Ticket sellers will use our service to get rid of any surplus tickets they might have to a show, sold out or otherwise.

(to be completed once buying and selling tickets runs smoothly) Ticket exchangers will use our service if they cannot make the showtime specified on their own ticket, but would like exchange their ticket for another to attend the desired show at a different time or on a different night. In this case,

the user chooses an option that indicates they only want to sell the ticket if they can purchase another ticket in return. In this case, this user will get a higher priority in our matching algorithm for matching with buyers for the tickets the user wants.

Section 4 ("Design")

Frontend: The frontend of the project will be implemented using standard technologies such as Bootstrap and jQuery, so that we can have dynamic, mobile-responsive content. We didn't think it was necessary to use more involved javascript frameworks such as AngularJS or React, but we will consider it if it becomes necessary. Currently, the idea is to use jQuery to make AJAX calls on the backend services.

Midware: We will be using Ruby on Rails for the backend code, as Henry and Richard have experience in Rails development. Ruby on Rails also makes deployment fairly straightforward.

We will create a RESTful service, and expose the API. This service provides functions to retrieve JSON data for events, prices, and individuals. As well, it provides functions to update and delete ticket orders and events. This service will be called by the front end.

Backend: The backend will be PostgreSQL. Rails abstracts the backend so we shouldn't have to worry about the sql flavor details, but PostgreSQL allows for easy deployment to Heroku.

Section 5 ("Timeline")

Bolded = Class Deadlines on the Project Page

March 27: **Project Status Website**

April 1: Get a frontend design layout for the front page of the web-app and profile page

April 1: Create Data model and general algorithm layout

April 5: Finish API implementation (back end)

April 8: Finish the integration between frontend and backend

Apri 14: **Prototype**

April 28: **Alpha Test**

May 5: **Beta Test**

May 8-12: **Demo Days**

May 14: **Final Documentation Submission**

Section 6 ("Risks and Outcomes") is the place to show that you've thought about what you need to do and what might go wrong or cause delay. We're not interested in generic risks like someone getting sick, but in perils specific to your plan: learning new languages, tools, and systems; dependence on data or software or hardware acquisition. What will you do if your preferred path is blocked? Give this some thought, so you don't discover a month from now that you simply can't have something that you need

Alex, Evelyn and Jason have little web-development experience so we foresee a sizable learning curve as we become accustomed to the environment (ruby on rails, javascript). So we will get the minimum done, that is, implementation for only buyer and seller features if we run into problems.