# Final Design Review:

## Proposed and Implemented Functionality:

In terms of end users, we implemented the User and Venue classes successfully. The User is designed to represent a customer looking to purchase event tickets or resell owned tickets. The Venue is designed to represent a host looking to sell event tickets.

We implemented the blockchain in full. We did so by building the previously proposed Transaction class, which stores the ticket being transacted and the transaction value, as well as the source and target of the transaction based on User and Venue ID's. Lists of Transactions are stored in a Block, which stores a Transaction in addition to a hash pointing to the previous Block in the Chain. Finally, the Chain class exists as a wrapper for Blocks pointing to each other.

For Users, we successfully implemented **Buy Ticket, List Ticket** and **Upgrade Ticket** for use cases involving ticket transactions. We implemented **Search** and **Explore** to allow them to find events of interest, and **Generate Ticket Code** for consuming tickets at an event they attend.

For Venues, we successfully implemented **Create Event** and **Manage Event** for adding and modifying events to our system. We also implemented **Schedule Release, Create Ticket** and **Manage Tickets** for issuing tickets, as well as changing their prices. Finally, we implemented **Validate Ticket Code** to allow a Venue to check the legitimacy of a ticket before approving its consumption at an event.

## Proposed and Unimplemented Functionality:

All proposed functionality was implemented. There were, however, some simple design changes we have accumulated and want to note. Our original conceptions of host and venue as separate classes did not serve our purposes. Rather, the functionality of both were combined into the current venue class. We originally suggested that all end users would be able to schedule release of tickets, but with the way in which list ticket is implemented for users who wish to put their tickets up for sale, only venues need to have the ability to schedule release.

## Additional Implemented Functionality:

We implemented skeleton code for a visualization of the blockchain that takes in placeholder variables, which can be seen in the last iteration of the code. However, as we mentioned in the README for the last iteration, since the function was not proposed in our initial design we were not going to have the visualization implemented. That said, since a note was made about it in the grading of the last iteration, we decided to address it again.

A few additional security aspects were added, including the triple check (consensus checking algorithm) between each of the two blockchains held by the venue class and event objects respectively, and the individual history held by each ticket in order to validate new transactions which occur, a robust generation of QR codes for each ticket, and a validation method for tickets when a QR code is scanned.

For explore, in addition to a simple weighting scheme that allows the program to intelligently recommend events to users, we took advantage of the python NLTK library to implement a basic natural language processing algorithm which divides up event description into salient tags (names and placenames) which can be used in addition to location and event name to keep track of user preferences over the course of their use of our software.

## Contributions:

Euirim Choi

**Iteration 1:**

Set up a Django application running with gunicorn on the request server in CSIL Wrote functions related to the **Buy Ticket** and **List Ticket** functionalities. Wrote a simple skeleton for the site, used throughout the site's pages.

**Iteration 2:**

Set up blockchain, request, and database servers on Amazon AWS. Ultimately, we ended up putting the blockchain server on a virtual machine in CSIL. Integrated S3 with Django's staticfiles (in order to serve them in production.

Set up a Django application running with gunicorn on the request server on EC2. Wrote most of the remaining views (those not implemented in iteration 1) for the venues, users, globals, and events apps of the project. These views serve templates to users requesting certain URLs. Wrote the skeletons for most of the templates which were later used in production-quality html/css integration.

Hussein El Kheshen

**Iteration 1:**

Developed the *Block, Chain,* and *Transaction* classes, which form the elements of our core blockchain functionality with Hayden.

Wrote **Most Recent Transaction** use case for traversing the blockchain.

**Iteration 2:**

Developed **Validate Ticket Code, Manage Tickets, Manage Event, Schedule Release**, and **Check Release** with Hayden.

Wrote **Venue Tickets** helper function to collect all tickets owned by venue for a given event.

Hayden Mans

**Iteration 1:**

Developed the *Block, Chain,* and *Transaction* classes, which form the elements of our core blockchain functionality with Hussein.

Designed blockchain security measures and wrote relevant hashing functions.

**Iteration 2:**

Developed **Validate Ticket Code, Manage Tickets, Manage Event, Schedule Release**, and **Check Release** with Hussein.

Developed read-write validation mechanism.

Added string representation of *Block* and *Transaction* classes to allow for deeper blockchain authentication.

Pablo Melana-Dayton

**Iteration 1:**

Set up a Flask application on the API server in CSIL.

Wrote API endpoints related to the **Buy Ticket** and **List Ticket** functionalities.

**Iteration 2:**

Completed the rest of the API endpoints that implement all blockchain functions.

Integrated the Django request server with the API endpoints. Wrote Django views with Euirim.

Ross Piper

**Iteration 1:**

Developed the *Trackers, User, Venue, Event, Seat*, and *Ticket* classes and the **Create Ticket, Buy Ticket, Upgrade Ticket, List Ticket** and **Generate Ticket Code** use cases with Ethan.

**Iteration 2:**

Developed the **Search**, **Create Tickets**, and **Create Event** use cases as well as the blockchain consensus mechanism.

Ethan Reeder

**Iteration 1:**

Developed the *Trackers, User, Venue, Event, Seat*, and *Ticket* classes and the **Create Ticket, Buy Ticket, Upgrade Ticket, List Ticket** and **Generate Ticket Code** use cases with Ross.

**Iteration 2:**

Developed **Explore, Update Preferences,** and **Chunk Tags** with Gina.

Samantha Rey

**Iteration 1:**

Developed HTML Wireframes for all of the functionalities being implemented in the first iteration in addition to a logo and front end form validation code and tests.

**Iteration 2:**

Developed Django Templating engine for the wireframes and deployed production level HTML/CSS as well as the blockchain visualization code.

Gina Yu

**Iteration 1:**

Worked on the front-end team to help design the Django application server.

**Iteration 2:**

Developed **Explore, Update Preferences,** and **Chunk Tags** with Ethan.