Allison Beasley & Joel Carlson

CS 412

April 15th, 2023

# CS412 Risks, Tasks, Timeline

## Risks – Main obstacles facing

Note: list of risks ordered from highest priority to lowest

* Supports Coin flipping
  + Highest priority for functionality
  + Will not take long to implement
  + Is a critical part
* Supports betting on Coin flip
  + High priority for functionality
  + Not very long to implement
  + Is critical
* Creation of database to hold player data
  + This will take a larger amount of time to complete
  + Requires default queries for operations to be created
  + following times rely on database
  + Is critical
* Supports user account creation
  + Requirement for functionality
  + Requires database creation
  + Upon creation of database, new entries will be easily added
  + Will not take long to include this functionality
  + Is critical
* Support use account authentication
  + Required for verification functionality
  + Follows user account creation
  + Requires database access and entry creation
  + Upon creation of database, new entries will be easily added
  + Will not take long to include this functionality
* Connects with database to allow for betting
  + High priority for functionality
  + Requires implementation of the DB
  + Longer development time
  + Is critical
* Support server-client relationship
  + Will require utilizing socket connection between the 2
  + Is a top priority of the project
  + Will not be too difficult to implement
  + Will not require an extended period to complete
* Multiple clients can participate at same time
  + Thread is created for each new connection
  + Utilizes socket relationship created in previous step
  + Will not be difficult to implement, but may have errors that are hard to detect
* Development of basic GUI for gameplay
  + For higher priority requirement usage, supporting buttons to eliminate terminal usage for game play
  + Displays up-to-date leaderboard
  + Will require a considerable amount of time to implement
  + Is critical for completion
* Development of GUI for user verification
  + Adapts console based verification for used.
  + Will not require much time to complete
  + Will be critical for completion
* Game play GUI displays current balance
  + Requires DB access/creation
  + Is important for usability
  + Will not be very hard to implement
* Game play GUI displays user name
  + ­is used as a verification of the User accessing the correct account
  + Requires DB access/creation
  + Will not be hard to implement
* Program ends upon closing window
  + Is very easy to implement­
  + Requires GUI of gameplay and verification to be work
  + Is not critical to functionality
  + Requirement for functionality
* Database can be updated concurrently
  + High priority for functionality
  + Requires game allowing for multiple threads for each user
  + Will not be time consuming to implement
* Supports leaderboard being updated concurrently
  + Requirement for functionality
  + Requires DB
  + Implementation requires separate “refresh data” thread for smooth implementation
* Allows user input for customized bet amounts within gameplay view of GUI
  + For betting requirements. Lower priority due to default bet amounts being implemented
* Game play GUI displays coin flip graphic
  + Low priority
  + Graphic will be an item to be completed at the end
  + Will take some time to add

## Tasks – What needs to get done; time for completion shown in timeline

Stage 1: General code structure: short time to implement

Fulfills: begins on all requirements

1. All classes created/initialized to allow for future development
   1. Created/initialized according to schema diagrams previously created
2. Functionality does not sever-client relationship, limited functionality

Stage 2: Connect server and client: medium time to implement

Fulfills: allows game play, server hosted game logic, and networked

1. Server implements gameplay
   * Method that randomly decides whether result is heads or tails
   * Communicates result to whether user wins or loses based on their selection prior to game play (will ultimately send result to client)
2. Server and client communicate
   * Client sends user selection of heads or tails (use temporary selections for now before GUI implementation)
   * Ability for server to send game play result to client
     + At this stage, print result to client console

Stage 3: Multiple clients can participate at same time: short time to implement

Fulfills: begins many users and keep track

1. Use threads to allow for multiple client connections between Client and Server
   1. As each client is added a new thread will be created for gameplay
   2. Will support any number of concurrent players

Stage 4: Database implementation: medium time to implement

Fulfills: keep track of players

1. Database created following database schema
   * Created within server’s Model class
2. Client access database through the server
3. Updates account balance based on game play result
   * Use temporary user until username is populated based on verification information
   * Use temporary default amount of $10.00 until user input supported

Stage 5: Supports user verification: medium time to implement

Fulfills: many users

1. Utilize differentiating statement for new and returning users in console
   * Same appearances will be shown to console in next two steps
   * Returning/creating will call different functions to update and/or verify in DB
   * Console version will be used for testing and pre-GUI work
2. Returning: Compares username/password to database through Server
   * Information request called by client sent to server which will query DB
   * Server returns relevant information to the client
   * Game will proceed if verification info is correct
     1. If not correct then error will be posted
   * Username/password can temporarily be entered via console until GUI developed
3. Support new user account creation
   * Function to add to the DB is called in client upon the click of enter
   * Game will be able to proceed if verification info is not a repeat
     1. If repeat username error will be posted

Stage 6: Supports leaderboard being updated concurrently: short time to implement

1. Upon action occurring function will be called to refresh the data being displayed
2. Data being refreshed now includes player leaderboard
3. All refreshing happens concurrently

Stage 7: Development of GUI for gameplay/user verification: long time to implement

Fulfill: Graphical interface for players

1. Create Controller class to connect all client GUI subclasses to Model
2. Create GameView class with radio buttons for heads/tails selection and button for playing
3. Display leaderboard with top 3 players within GameView class
4. Create GameViewController class with NewGoButtonActionListener and PlayButtonActionListener classes to support button functionality and connect to Controller
5. Create InitialView class with buttons for new or returning users
6. Create InitialController class with NewButtonActionListener and OldButtonActionListener classes to support button functionality and connect to Controller
7. Create VerificationView class with textfields for username/password and a button for submitting
   1. Ability to display error message upon username/password verification failure
8. Create VerificationController class with SubmitButtonActionListener class to support button functionality and connect to Controller
   1. Via Controller class receive info whether username/password matched with a user in the database. If not, send notification of failure (likely via Boolean) to VerificationView
9. Add ability for GUI to change between InitialView, VerificationView, and GameView displays upon successful submissions in InitialView and VerificationView classes

Stage 8: Allows user input for customized bet amounts within gameplay view of GUI: medium time to implement

Fulfill: none, continued perfection of allowing gameplay

1. Within GameView class implement TextField with betAmount variable
2. Connect NewGoButtonActionListener to betAmount variable
   1. Will update model through controller
3. Betting amount will now utilize amount passed in textfiedl

Stage 9: Program ends upon closing window: very short time to implement

Fulfill: game connection closing upon window closing

1. Disconnect client from server upon closure of gameplay window

Stage 10: Game play GUI displays username, current balance, and coin flip graphic: medium time to implement

Fulfills: none, continued perfection of allowing gameplay

1. Create label to hold usename
2. Create label to hold current balance in dollars
3. Create label to hold picture of coin

Stage 11: Test for edge cases and ensure lack of bugs: varying time to implement

Fulfills: none, continued perfection of allowing gameplay

1. Test for entering blank usernames, incorrect entries for betting amounts, etc

Stage 12: Fix any formatting, make prettier as desired: medium time to implement

Fulfills: none, continued perfection of allowing gameplay

1. Arrange GameView in a way that best presents information while balancing functionality

## Timeline – timing for sprints

* Sprint 1: Friday April 21st, 11:59 pm
  + Completion of stages 1-3,
  + Dependencies:
    - All stages dependent on completion of stage 1
    - Stage 2 dependent on stage 1
    - Stage 3 dependent on stage 2
* Sprint 2: Friday April 28th, 11:59 pm
  + Completion of stages 4-8
  + Dependencies:
    - Stage 4 dependent on stage 2 for general client/server connection structure
    - Stage 5 dependent on stage 3
    - Stage 6 on stage 5
    - Stage 7 on all previous
    - Stage 8 on all previous
* Sprint 3: Thursday May 4th, 11:59 pm
  + Completion of stages 9-10, started stages 11-12
  + Dependencies:
    - Stages 9-10 dependent on all previous
    - Stage 11 dependent on completion of all previous
    - Stage 12 depends on use but is dependent on stage 7

DUE DATE: MAY 5TH BEFORE CLASS