

***Note that you can refine your testing plan as the project development goes. Keep the change log as follow:***

*ChangeLog*

| **Version** | **Change Date** | **By** | **Description** |
| --- | --- | --- | --- |
| 1 | 2025/02/07 | Caelan | Removed Extra Writing |
| 2 | 2025/02/26 | Niko, Owen, Caelan, Cole | Added Tests to test methodology |
| 3 | 2025/03/17 | Cole | Added Roulette Tests test methodology |
| 4 | 2025/03/27 | Caelan, Niko | Added documentation for security scan and load testing |

# Introduction

## Scope

The tests will encompass testing the individual games of our website, namely the logical features like betting, logging in, and selecting games

## Roles and Responsibilities

Detailed description of the Roles and responsibilities of different team members like. Note you only need to list the role you have in your team. There are some example roles.

* Team Lead
* QA
* Configuration Manager
* Database Manager
* Designer
* Developer

| **Name** | **Net ID** | **GitHub username** | **Role** |
| --- | --- | --- | --- |
| Caelan Myskiw | myskiwc | Naleac | Team lead, developer, QA |
| Owen Zonneveld | zonneveo | TheOfficialOzone | Configuration Manager, developer |
| Michael Walld | walldm | walldmtd | Database Manager, developer, Designer |
| Cole Chuchmach | chuchma1 | Cole-Chuchmach | Developer |
| Niko Christie | christ37 | NikoChristie | Developer, QA |

# Test Methodology

## Test Levels

**Test Levels define the Types of Testing to be executed on the Application Under Test (AUT**). In this course, **unit testing, integration testing, acceptance testing, regression testing, and load testing** are mandatory. Please describe how you will do these testings. You may skip load testing at this moment. Please revisit it after the related lecture is given.

Unit Tests:

| **Feature Name** | **Tests** |
| --- | --- |
| Horse Race Betting | Verify placing the various types of bets (Show, Place, Straight), verify betting & multi-betting on multiple horses, verify wager payout is working for all types of bets (Show, Place, Straight), verify wager does not payout when it shouldn’t for all types of bets, verify losing money on bad bets, verify destructions and creation of new horses, verify uniqueness of generated horses to one-another and pre-existing horses, verify betting & multi-betting on generated horses |
| Roulette | Verify user can login, and progress to the roulette page from the lobby, verify that the users balance appears on the roulette page, verify that the roulette countdown timer appears on the roulette page, verify that the roulette wheel appears on the roulette page, verify that the roulette board appears on the roulette page, verify that the user can select different bet amounts, verify that the user can place a bet on each cell type (*48 different choices*), verify that the user cannot bet more than their current balance, verify that a valid bet that wins pays out, verify that a valid bet that loses doesn’t payout. |
| User Accounts | Verify users are created with valid parameters, verify users are not created with missing parameters, verify users are not created with invalid parameters, verify users are not created with duplicate username/email, verify user account balance is positive and properly formatted, verify user account balance can be modified, verify pages redirect if user not signed in |
| Casino Lobby | Verify under construction tables exist in the lobby, verify horse racing table exists, verify under construction roulette table exists, verify under construction slots exists, arrive at casino lobby successfully, test logging out brings you to the new session page, verify horse racing button brings you to the betting/racing page. |

Integration Tests:

| Test | Description |
| --- | --- |
| Verify Wallet | Verify the wallet associated with an account can be used for betting |
| Verify Database | Verify that the data in the database is accurately displayed to user |

Regression Tests:

| Test | Description |
| --- | --- |
| Full testing suite | Every Pull request should run the full testing suite to ensure that all code meets QA standards. |

Acceptance Tests:

| Test | Description |
| --- | --- |
| Verify accounts | Test that you can navigate to a login page, create an account, and login. |
| Verify horse race betting | Test that you can navigate to a horse race and place a bet on a given horse. Verify that after the race the user is paid out the correct amount. |
| Verify horse race reset | Verify that after a race has finished, the horses move back the the starting line and race again |
| Verify roulette | Verify that you can create a new user account or login using an existing account, then navigate to the roulette game page. After, place a bet on any cell type that you like and verify that the amount paid out matches what should be paid out given the winning number and your bet type. |

Load Tests:

| Test | Description |
| --- | --- |
| Account System (Login) | Logs into a user’s account so they can place bets, 20x in total |
| Horse Bet & Balance | Places 1 bet on a horse, (this tests a user’s balance as well) we do this 5x per user |
| Roulette Bet & Balance | Places 1 bet on red, (this tests a user’s balance as well) we do this 5x per user |
| Chat | Send 1 message in chat, do this 2x per user |

Requirements:

* List the class/method/core feature you plan to test and how you would like to test them and its acceptance criteria.
* For unit testing, at least 10 unit tests for EACH core feature to cover the code related to each core feature
* For integration testing, at least 10 in total to cover core features.
* Acceptance testing for each core feature. Let’s use an end-user test for this. You can ask real end-users or your team members to go through each user story and see if the requirements are met.
* For regression testing, you need to execute all above unit tests + integration tests you have for each commit pushed to the main branch.
* For load testing, when designing the load, make sure at least two requests associated with every core feature are included in the test load.

## Test Completeness

Here you define the criterias that will deem your testing complete. For instance, a few criteria to check Test Completeness would be

* 100% back-end code coverage (mandatory for this project), all the back-end source code should be covered by test cases.
  + This is defined as full code coverage of rails controllers, models, and helpers
* Load test 20 users with a total 200 requests per minute

# Resource & Environment Needs

## Testing Tools

Make a list of Tools like

* Github Issues
* Ruby on Rails
* Automation Tools
* Bundler
* Brakeman
* JMeter

## Test Environment

It mentions the minimum **hardware** requirements that will be used to test the Application.

Example, following **software’s** are required in addition to client-specific software.

* MacOS Catalina 10.15 or newer
* Ubuntu Jammy 22.04 or newer
* Windows 11 or Windows 10 version 2004 and higher
* Ruby
* MySQL
* GitHub Actions
* Rails framework (Ruby on Rails) version 8 or higher
* …

# Terms/Acronyms

Make a mention of any terms or acronyms used in the project

| **TERM/ACRONYM** | **DEFINITION** |
| --- | --- |
| API | Application Program Interface |
| AUT | Application Under Test |