Bachelor of Information Technology

Advanced App Development

Level 7, Credits 15

Assessment Task

Project

Assessment Overview

This is the main assessment for this course and is worth 80% of the overall mark. Your individual mark is determined by your effort and input into the final deliverable. You will submit a completed and deployed software project.

Assessment Table

Assessment Activity	Weighting	Learning Outcomes
Practicals	20%	all
Project	80%	all

Conditions of Assessment

There will be time at the end of semester to work on the project during class time. However, you will need to do more work during your learner-managed time. You are to produce a project satisfying all the user stories. You will submit source code to GitHub and provide links to all associated services (production link, etc.).

This assessment needs to be completed and submitted by the last day of semester.

Authenticity

This is an individual assignment and therefore must be your own work. We encourage the use of third-party libraries/frameworks, but make sure that you can legally use them. Look for LGPL, MIT or Apache license. For example, GPL may have some constraints that you need to consider. Plagiarism, outsourcing or unauthorised code use will not be tolerated and may result in a 0 mark.

Late Submission, Reassessment, Extensions

The School process in relation to Submissions, Extensions, Resubmissions and Resits complies with Otago Polytechnic Policies. Students can view policies on the Otago Polytechnic Website located at http://www.op.ac.nz/.

Resubmission is where an original assessment is returned to the student for minor reworking and then being resubmitted for final grade. Where a student achieves a D grade for any assessment, an application for resubmission may be made to the Head of School. A maximum of two resubmissions will be permitted in any one year for any student.

Resubmissions are completed within a short time frame (usually no more than five working days) and usually must be completed within the timing of the course to which the assessment

relates. Resubmissions will be available only to students who have made a genuine attempt at the first assessment opportunity. The maximum grade awarded for a resubmission will be C-.

Information about late submission and extensions can be located in the Course Outline, available on the course Moodle page.

Learning Outcomes

At the successful completion of this course, students will be able to:

1. Critically evaluate and implement a range of programming paradigms.
2. Create efficient full-stack applications using advanced industry techniques, tools and frameworks.

Instructions

This is a project assessment. You will develop a multi-player online poker game. It will be a Progressive Web App built on ReactJS and Firebase.
Rules and description of poker are available online, e.g.:
https://www.considerable.com/entertainment/card-games/basic-poker/

User Stories

As a user, I need to create a new game so others can join.
As a player, I want to join a game that is waiting for players.
As a game owner, I want to start the game when enough opponents joined the game. After the game is started, nobody can join any more.
Each player will start with 5 cards.
As a player, I will get one chance to exchange my chosen cards with new ones from the deck.
As a player, I need to be able to see who's turn it is.
As a player, I want to see who won the game (scoring of Poker is widely known/available).
As a player, I need to be able to run the game and make my turn offline.

In other words, this is a very simplified poker. There's no "betting" phase, no raise/call/pass actions. The "gambling" element has been removed.

Bonus stories

Feel free to add the whole Poker experience (betting and taking turns until every bet is "called" or players "fold").
As a game owner, I need to add AI/computer players to fill spots of friends that I don't have.
Strategies for the AI/Computer players (e.g. bluffer, risker, conservative, etc.) will be chosen randomly, so I'm not able to predict my opponent's behaviour.
Consider an external service for the AI/Computer players (via REST API, or a Cloud Function).

Technical Notes

Feel free to use any 3rd-party frameworks or libraries. Make sure that the licenses allow you to use the library for your purpose.
One exception is that I would like to see your own implementation of the game setup and poker scoring and the AI/Computer player implementation. Basically, don't use a third-party poker game and submit it as your work.

Submission Instructions

All source code must be committed to a given GitHub repository. There must be regular commits throughout the project implementation phase to show your progress. One big commit at the end is dangerous and unprofessional and may be treated as suspicious. Claims of lost code at the end of the semester will not be accepted.
Please put link to the deployed production version in a readme file in your project repository.

Marking Schedule

Component	Weight
Architecture (classes, usage of design patterns, etc.)	40%
Code quality (code readability, formatting, commenting)	20%
Automated tests	25%
User Experience	15%

The score will be multiplied by the amount of functionality (user stories) that were implemented.

	10-9	8-7	6-5	4-0
Architecture	System is functionally complete. There's no code duplication and components are designed for reusability and widely reused. Design patterns are chosen correctly for the right scenarios and implemented correctly. Anti-patterns are not present in the code. Robust security measures designed and implemented.	System is mostly complete. There are very few minor bugs. Architecture designed for code reuse, but there are small areas of anti-patterns. Design patterns considered and implemented. Security considered and implemented for most common cases.	Important functionality is complete and works without obvious failures. Some successful attempts at code reusability and prevention of duplicate code. Some design patterns applied correctly, but there are also anti-patterns present. Security considered and partly implemented.	System is incomplete or not working correctly. No apparent code reuse or lot of duplicate code. Design patterns not considered or not applied correctly. Anti-patterns are apparent throughout the system. System is insecure.
Code Quality	Code perfectly formatted, easily readable. Comments explain the thinking behind the code. It is clear what each class/method does and code blocks are kept to manageable amount.	Code well formatted and organised with areas for improvement.	Code is mostly well organised, but there are some parts that are hard to interpret. Comments are missing or not in useful places. Some files/classes or methods are too long or too nested.	Poorly formatted code, hard to interpret.

	Files kept to manageable size.			
Automated tests	Comprehensive, robust and reliable test suite, including tests for security rules. Tests run repeatedly with the same result. Tests all “happy path” scenarios as well as exceptional cases (e.g. denied authorisation, etc.). Source code follows the same high standards as the main production code.	Tests cover most of the functionality and are reliable. Mostly “happy path” scenarios are tested or there are some superfluous tests. Source code mostly follows the standards for the main production code.	Important parts of the system are tested with automated tests. Tests are mostly reliable.	Tests don’t test the important functionality. Source code is messy, duplicate and with apparent anti-patterns.
User Experience	System is easy and pleasant to use and visually very appealing and consistent throughout the system.	System is mostly easy to use with some slightly unintuitive areas. Visually appealing but there are some inconsistent areas.	System can be used without undue effort.	System is hard to use an unintuitive and visually unappealing.

0 marks = nothing done.