Appendix I:

Test Plan

Technopoly
Test Plan

PREPARED FOR

Customer: Technopoly

PREPARED BY

Group 3

Version History

Document History

Version	*Date*	*Author*	*Description of Change*
1.0	18/02/19	Colette Casey	Draft
2.0	11/03/19	Colette Casey/Ismael Florit	Revised

Approvers List

Name	*Role*	*Approver/Reviewer*	*Approval/Review Date*
Ismael Florit	Development Team	*IF*	12/03/19
Barbara Murtland	Development Team	*BM*	12/03/19
Dave Kennedy	Development Team	*DK*	12/03/19
Timothy Lewis	Development Team	*TL*	12/03/19

Reference Documents

Version	*Date*	*Document Name*
10th Edition	*Semester 2 2019*	Software Testing – Ian Sommerville
Lecture Notes	18/02/19	PowerPoint Chapter 12: Software Verification – Testing Strategies

Table of Contents

1. Introduction

1.1 Project Overview

This Test Plan prescribes the scope, approach, resources, and schedule of all testing activities for the Technopoly project.

The plan is to identify:

- Test strategy
- Items and features to be tested
- Types of testing to be performed
- Risks associated with the plan
- Test criteria
- Resources and schedule required to complete testing
- Test deliverables

2. Test Strategy

The group 3 test strategy shall be defined by Test-Driven Development (TDD). As such, the group seeks to “Interleave testing and code development” (Beck 2002; Jeffries and Melnik 2007) and will be influenced by the V-Model for test implementation.

The specification shall be used to identify the features to test to meet the customer's core requirements. From the list compiled, suitable JUnit tests will be created. The aim is to design tests to be carried incrementally. As the code evolves, these can then be used as automated tests for regression testing. The aim is to have repeatable, quantifiable, and measurable testing activity. Areas suited to exploratory testing will also be identified – for when a more fully integrated, working system is in place.

2.1 Scope of Testing

Key Assumptions

- Exploratory Testing is to be carried out as segments and iterations of the build are ready for testing.
- Non-Functional requirements will primarily be considered during exploratory testing.

2.2. Features to be tested

Based on the customer’s software requirements specification, these are the features of the Technopoly project identified for testing:

- Game has up to four players.

- Players take turns.
- Players throw two virtual dice – number range between 1–6.
- 'InvestNI' square present – equivalent of 'Pass Go', includes crediting resources.
- 'Runway' square present – equivalent of 'Free Parking.'
- Four fields exist:
  - Two fields with three areas.
  - Two fields with two areas.
- Test for ownership whole field – allows development.
- Test an area can be developed, even if player is not in the area.
- Three software developers are hired before a CTO can be hired – equivalent of a hotel.
- Player taking a turn is told what their obligations and opportunities are.
- If player resources change, the system indicates the reason and announces the new balance.
- Player can interact with the menu to indicate their choice of action.
- If player lands on a space owned by another player, they must pay a license fee.
- When one player runs out of resources, the player with most resources is declared the winner.
- If player no longer wants to play, they can terminate game.
- When game ends, the amount of resources each player holds is shown.

2.3 Features Not To Be Tested

These features are not be tested because they are not included in the software requirement specification:

- User Interface
- Hardware Interfaces

The customer requirements are for a simple console based game – conducted in natural language only.

2.4 Test Types

The Technopoly project will mainly focus on Development Testing, carried out by members of the team developing the system as the software becomes available to test.

The 3 main types of testing to be conducted are:

- **Unit Testing** – Using the JUnit framework. The process of testing methods, objects and classes with a goal to confirm each subsystem is correctly coded and carries out the intended functionality.

- **Integration Testing** – Individual software modules are combined and tested as a group. The goal is to test the interfaces between the subsystems.

- **System Testing**: Conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

Due to the risk of time constraints, System testing will mainly be carried out in the form of exploratory testing, alongside regression testing.

User Testing and Acceptance Testing will be planned with the customer in the later stages of development.

2.5 Risk and Issues

Risk	Mitigation
Team members are new to JUnit testing and Java development.	Continue to attend Software Engineering and Programming lectures on this subject.
The project schedule is extremely tight and it will be hard to complete project on time.	Set Test Priority for each test activity if time becomes an issue.

3 Test Criteria

3.1 Suspension Criteria

If team members report there are 50% of test cases failed, testing will be suspended until the development team resolves all the failed test cases.

3.2 Exit Criteria

Run rate should be 100%, unless a clear reason is given and a plan for resolution put in place.

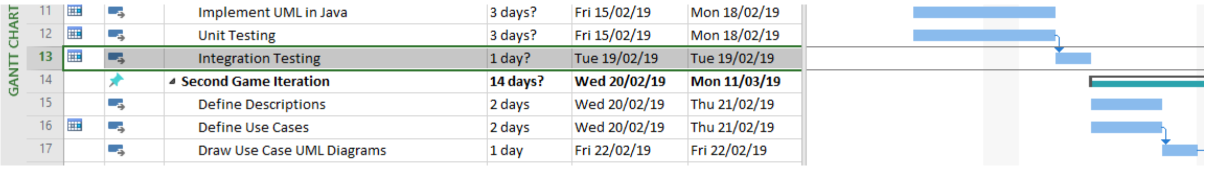
4 Resources & Schedule

4.1 Human Resources

Member	Tasks	Personnel	Estimated Effort
Test Manager	Create and define test plan.	Colette Casey	5 hours
Developers in Test	Implement the Unit tests, Integration tests, System tests, test suite etc.	Ismael Florit Dave Kennedy Timothy Lewis Barbara Murtland Colette Casey	5 hours 5 hours 5 hours 5 hours 5 hours

4.2 Schedule to Complete Tasks

Draft 1 of project Gantt Chart shows how Group 3 scheduled Unit and Integration testing during early project planning:



11	Implement UML in Java	3 days?	Fri 15/02/19	Mon 18/02/19
12	Unit Testing	3 days?	Fri 15/02/19	Mon 18/02/19
13	Integration Testing	1 day?	Tue 19/02/19	Tue 19/02/19
14	**Second Game Iteration**	**14 days?**	**Wed 20/02/19**	**Mon 11/03/19**
15	Define Descriptions	2 days	Wed 20/02/19	Thu 21/02/19
16	Define Use Cases	2 days	Wed 20/02/19	Thu 21/02/19
17	Draw Use Case UML Diagrams	1 day	Fri 22/02/19	Fri 22/02/19

5 Test Deliverables

These are the test deliverables to be provided:

- Test Plan document
- Regular team reporting on test results and defect detection at daily scrum meetings – fixes to be discussed, planned and implemented by relevant members of the team.
- Screen shots of defects to be taken during exploratory testing to help with debugging and planning solutions.

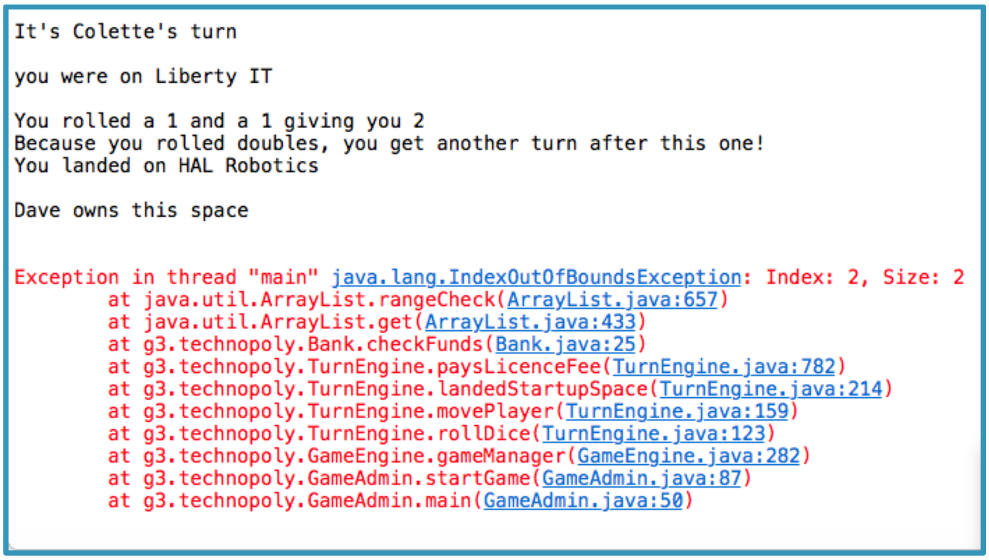
5.1 Test Results Table

No.	Test Name	Category	Status
1	Game has up to four players	JUnit Testing	Pass
2	Players take turns	Exploratory	Pass
3	Players throw two virtual dice – number range between 1–6	JUnit Testing	Pass
4	'InvestNI' square present – equivalent of 'Pass Go', includes crediting resources	JUnit Testing	Pass
5	'Runway' square present – equivalent of 'Free Parking'	JUnit Testing	Pass
6	Four fields exist	JUnit Testing	Pass
7	Two fields with three areas	JUnit Testing	Pass
8	Two fields with two areas	JUnit Testing	Pass
9	Three software developers are hired before a CTO can be hired – equivalent of a hotel	JUnit Testing	Pass
10	Test for ownership whole field – allows development	Exploratory Testing	Pass
11	Test an area can be developed, even if player is not in the area	Exploratory Testing	Pass
12	Player taking a turn is told what their obligations and opportunities are	Exploratory/ Integration Testing	Pass
13	If player resources change, the system indicates the reason and announces the new balance	Exploratory Testing	Pass

14	Player can interact with the menu to indicate their choice of action	Exploratory Testing	Pass
15	If player lands on a space owned by another player, they must pay a license fee	Exploratory Testing	Pass
15	When one player runs out of resources, the player with most resources is declared the winner	JUnit Testing	Pass
17	If player no longer wants to play, they can terminate game	Exploratory Testing	Pass
18	When game ends, the amount of resources each player holds is shown	Exploratory Testing	Pass

5.3 Example of Screen Shot Used for Debugging

Screen shots were taken when defects found during exploratory testing to help replicate and resolve issues found:



```
It's Colette's turn

you were on Liberty IT

You rolled a 1 and a 1 giving you 2
Because you rolled doubles, you get another turn after this one!
You landed on HAL Robotics

Dave owns this space


Exception in thread "main" java.lang.IndexOutOfBoundsException: Index: 2, Size: 2
        at java.util.ArrayList.rangeCheck(ArrayList.java:657)
        at java.util.ArrayList.get(ArrayList.java:433)
        at g3.technopoly.Bank.checkFunds(Bank.java:25)
        at g3.technopoly.TurnEngine.paysLicenceFee(TurnEngine.java:782)
        at g3.technopoly.TurnEngine.landedStartupSpace(TurnEngine.java:214)
        at g3.technopoly.TurnEngine.movePlayer(TurnEngine.java:159)
        at g3.technopoly.TurnEngine.rollDice(TurnEngine.java:123)
        at g3.technopoly.GameEngine.gameManager(GameEngine.java:282)
        at g3.technopoly.GameAdmin.startGame(GameAdmin.java:87)
        at g3.technopoly.GameAdmin.main(GameAdmin.java:50)
```