TEST PLAN

GuessWho!™

Face-Name Matching Game for Dementia Patients

Delivered by

Bookies

Joshi Chaitanya Krishna(U1522971F)

Genevieve Lam Wen Qi (U1521863H)

Sharma Vidur (U1522940D)

Yong Chen Feng (U1620913B)

Heng Zhi Guang (U1620660F)

Chen Guanyu (U1621093D)

Submitted to –

Dr Shen Zhiqi

Nanyang Technological University

School of Computer Science and Engineering

Revision History

Version #	Date	Author(s)	Comments
1.0	8/11/2018	Bookies	Preliminary Version

Table of Contents

1 Master Test Plan

1.1 Test Plan Identifier

Test Plan Identification Number: MTP_01 – GuessWho!™

1.2 Introduction

1.2.1 Purpose

The purpose of this Test Plan is to document the process of testing that Bookies will follow.

This Test Plan supports the following objectives:

- Identify the areas of testing
- Identify the motivation and idea behind the test areas to be covered
- Describe the testing methodology used
- Identify the required resources and budget available for comprehensive testing

1.2.2 Scope

This test plan describes the unit, subsystem integration, and system level tests as well as other tests that will be performed on components of GuessWho!™. Prior to the testing of each subsystem, the assumption that each of them has undergone an informal peer review is being made. Unit tests will be done through a test driver program that performs boundary checking and basic black box testing. Subsystem tests will verify defined interfaces of modular components of the application:

- Login and signup activity
- Caregiver views
- Face recognition game
- Scores

The external interfaces to be tested:

- RoboMongo database interface

The following performance measurements will be tested:

- Response time for successful submission of answer to database
- Response time for successful displaying result in the webpage
- Response time for game progression

1.3 Test Items

Functional requirement stated in the System Requirement Specification.

- Version: System Requirement Specification v1.0

Non-functional requirement stated in the System Requirement Specification.

- Version: System Requirement Specification v1.0

Use Case Description

- Version: System Requirement Specification v1.0

Test cases

- Version: Test Cases and Requirements Test Coverage Report v1.0

1.4 Features Tested

All features listed below are to be tested:

- Functional requirement stated in the System Requirement Specification (SRS)
- Non-functional requirement stated in the System Requirement Specification (SRS)
- User acceptance testing
- All the use cases developed to describe the system usage.

1.5 Features Not Tested

Due to time constraints, the completion of the planned tests will depend on the status of implementation. Since GuessWho!™ is a prototype that showcased only the face recognition game functionalities, other components that had yet to be implemented will not be tested. However, Bookies' QA Manager & Engineer will still generate the test cases for selected unimplemented components.

1.6 Test Approach

Manual unit testing approach is used due to time constrain encountered in this project. Appropriate unit tests must be conducted and passed before any code can be checked into the SVN. Any issues found must be brought forward to a common pool of bugs along with its

details including the person who discovered the bug and the severity of it. The bug is then assigned to a developer who will then fix the issue. Periodical retesting of code is done to ensure that all code uploaded is bug-proof. All the information obtained, including cause and solution to the bug, is recorded by the QA Manager & Engineer for future references.

1.6.1 Testing Techniques and Types

1.6.1.1 Unit Testing

Unit testing is performed to verify all functional requirements are met successfully.

Table 1 Unit Testing

Technique Objective	Verify system functional requirements
Technique	Verify the requirements put forth in the use cases are met
Oracles	GuessWho!™ Use Cases
Success Criteria	The following are successfully tested: ● All key use-case scenarios ● All key features

1.6.1.2 User Interface Testing

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the UI provides elderly dementia patients with minimal and appropriate access and navigation through the functions of target-of-test. In addition, UI testing ensures that the objects within the UI function as intended to corporate or industry standards, and follow standards, conventions and metrics stated in the Quality Plan document.

Table 2 User Interface Testing

Technique Objective	Navigation through the unit of application to be tested showing correct screens for different inputs during Face Recognition games, doing correct computation of scores, displaying different accounts and personal collections for different users logged in

Technique	Create tests which corresponds to different scenarios of users usage
Oracles	The tester will verify proper functionality based on requirements
Required Tools	Personal Computer
Success Criteria	All tests can be carried out without application termination error and screens for each scenario show up as expected

1.6.1.3 System Testing

The objective of System Testing is to study the interaction between functions within the entire integrated system with end-to-end flow. It also verifies whether the whole system meets the specified requirements, both functional and non-functional.

Table 3 System Testing

Technique Objective:	Integration Test has been completed. All the individual modules have been integrated into a single application: GuessWho!™.
Technique:	The functional testing shall be conducted through an organized testing flow. A testing flow will be followed to test the integrated system to ensure all functionalities work seamlessly together. This is an integral step to ensure full functionality of the application as modules working individually might produce errors when integrated together. References to the System Requirement Specification will be made to ensure all functional requirements are met.
Oracles:	GuessWho!™ Use Cases GuessWho!™ System Requirement Specification
Required Tools:	Personal Computer
Success Criteria:	The following criteria must be met for functional testing:

	• 100% of functional requirements covered • 100% of basic paths covered • 100% of common usage scenarios covered

1.6.1.4 Performance Testing

Performance testing will be carried out to test response times and other time-sensitive requirements are measured and evaluated. The goal of Performance Testing is to verify that performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune a target-of-test's performance behavior as a function of conditions such as workload or hardware configurations.

Table 4 Performance Testing

Technique Objective:	Normal anticipated workload Anticipated worst-case workload
Technique:	Upserting to the database for photo uploading (as base64 string) from the application to MongoDB and registration purposes. Executing MongoDB query for retrieving login details, photo paths and patient information in caregiver menu from MongoDB to application.
Oracles:	Software Timers
Required Tools:	Personal Computer installed with Node.js
Success Criteria:	The test completes within the time indicated by the use cases

1.6.1.5 Stress Testing

Stress testing is a series of tests to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. Stress testing is usually performed by simulating scenarios with low resources, poor network connectivity and pushing the system to unexpected scenarios.

Table 5 Stress Testing

Technique Objective:	Multiple users accessing database from their own laptop or PC Running on a system with below recommended requirements (e.g.: limited internet speed)
Technique:	To test limited resources, tests should be run on a single machine with RAM and persistent storage space on the server being reduced or limited when making queries against the database under heavy load.
Oracles:	Determined by the tester
Required Tools:	Personal Computer
Success Criteria:	The ability to access the MongoDB database in under 1 second at each workstation

1.6.1.6 Configuration Testing

Configuration testing verifies the test-target operates correctly under different software configurations and how it interacts with different software.

Table 6 Configuration Testing

Technique Objective:	Verify the test-target functions correctly on different platforms and under different configurations
Technique:	Run other applications on the testing PC while running GuessWho!™ application. Run GuessWho!™ application on several brands and types of PC or laptop with different system specification and operating system. Carry out the same actions for the website on different devices.

Oracles:	Test-target behaviour
Required Tools:	Brands: Multiple Brands of PC/Laptop Operating System: Windows, Linux, Macintosh
Success Criteria:	The application behaves in the normal flow

1.7 Entry and Exit Criteria

1.7.1 **Test Plan Entry Criteria**

Coding of the application has completed, and the team has informally reviewed the code.

1.7.2 Test Plan Exit Criteria

The project's deadline is due, all the use case requirements have been verified, or the deadline for virtual server rental is up.

1.8 Suspension and Resumption Criteria

Testing will be suspended when encountered with critical design flaws that will require an application interface redesigned. Testing will resume when amendments to the code is completed and reviewed.

1.9 Deliverables

The following artifacts shall be provided as testing deliverables available to the stakeholders.

1.9.1 Test Evaluation Summaries

Test Evaluation summaries will outline the tests conducted and their results. Debugging methods and outcome will also be recorded. In the event where debugging is unsuccessful, alternative solutions will be proposed in the summaries.

1.9.2 **Incident Logs and Change Requests**

Incident log entries will be made for all bugs found during testing. The log will also be used to track the status of the bugs. Any modifications to the requirements must be done

through change requests, which will ensure that the proposed change is fully reviewed before being incorporated into GuessWho!™.

1.10 Environmental Needs

This section contains the non-human resources needed for the Test Plan.

1.10.1 Base System Resources

The following sections list the resources required for the implementation of GuessWho!™.

1.10.1.1 Network

In order to facilitate information transfer between the application on remote device and the virtual server, a connection must be established between the virtual server and the device running the application. As such, the device must be able to access NTU's network.

To provide an NTU VPN encrypted network connection for remote mobile devices, laptops already connected to NTU VPN can be set up as Wifi hotspots.

1.10.1.2 Laptop and PC

A PC/Laptop running Windows or Mac OS

1.10.1.3 Production Support Systems

- To maintain backend data, NTU virtual server computers are required
- Server computers must have at least
  - 4 Gigabytes of memory space
  - 10 Gigabytes of free space
  - X86, Dual Core 2.0GHz clock speed system architecture

1.10.1.4 Operating Systems

Windows 7 or higher or Mac OS 10 or higher

1.10.1.5 Server System

Windows XP and above.

For backend operations, Mongodb version 3.4 and above.

Node.js version 6.10 and above.

1.10.2 Additional Testing Resources

Since the team members will be completing most of the work on their laptops, software needed for this project will include:

- Sublime Text Editor
- Robomongo
- Chrome browser

1.11 Responsibilities, Staffing and Training Needs

This section outlines the personnel necessary to successfully test GuessWho™ web application.

1.11.1 People and Roles

Table 8 Master Test Roles and Responsibilities

Role	Responsibilities
Project Manager (*Chaitanya Joshil*)	Identify test ideas to be conducted, Define details of tests to be conducted, Document change requests, Review test results with QA Manager
QA Manager & Engineer (*Chen Guanyu*)	Define test approach, Define test automation architecture, Verify test techniques, Define testability elements, Organize test implementation, Evaluate test results and report to Project Manager
Lead Developer (*Heng Zhi Guang*)	Support the administration of test data and the database.
Front-End Developer (*Genevieve Lam*)	Define test classes required to support front-end testability requirements defined by the test team.
Back-End Developer (*Yong Chen Feng*)	Define test classes required to support back-end testability requirements defined by the test team.

Release Engineer (*Vidur Sharma*)	Ensure major components in the GuessWho!™ application are tested thoroughly. Demonstrate the flow of application to the testing team during release. This allows testing to be done, making sure that the application can be showcased smoothly.

1.11.2 Staffing and Training Needs

For this project, the number of staff is fixed throughout. As such, most of the staff will assume some form of testing role. Training will be customized to each of the group member's strengths and weaknesses to accommodate differences in abilities.

1.12 Risks, Dependencies, Assumptions, and Constraints

Table 9 Master Test Risks and Constraints

Risk	Mitigation	Strategy Contingency
Test cases designed are not comprehensive	Check through every test case for adequacy before submission.	Ensure test data is redefined. Ensure Test Plan is reviewed and modified. Ensure test cases are re-written and reviewed by the team.
Limited coverage of functionality as insufficient test cases were designed	Gather the entire team and examine the list of items to be tested in order to identify test cases that were not designed.	Ensure that test cases are written according to different aspects of GuessWho!™ to ensure full coverage
Poor time management	Implement a schedule to carry out testing. Ensure that there are consistent reviews and deadlines are adhered to.	Make sure high priority test cases are completed first.

1.13 Approval and Signoff

The initial test plan must be approved by the Project Manager. Upon completion of such tests, the testers will sign off use case requirements with the Project Manager's consent.

2 Integration Test Plan

2.1 Test Plan Identifier

Test Plan Identification Number: ITP_01 – GuessWho™

2.2 Introduction

2.2.1 Objective

The necessary tests to ensure all the elements of GuessWho!™ are appropriately integrated are described in the Integration Test Plan. In order to check that the unit-tested modules interact with each other appropriately, integration testing is carried out.

2.2.2 Scope

The process of integration testing as well as the roles of the members of the Bookies team and their stakeholders will be described in this document.

2.3 Test Items (Function)

Integration testing will be performed periodically as software components are integrated into the system. To test the GuessWho!™ application and website, a bottom-up approach will be used. In this document, the integration tests described are at the software component level.

2.4 Features Tested

The integration of each component will be tested using a driver when carrying out Integration Tests. In order to integrate components of higher levels, the testing team must successfully complete the integration of lower components.

2.5 Features not Tested

Features that

- are not a part of this release of the application
- have low risk and have been previously verified as stable
- are released not as a functional part of the current release version of the software

2.6 Approach Strategy

Together with the development team, the QA Manager & Engineer and Lead Developer will perform the Integration Test whenever necessary. In order for a program to enter the Integration Test environment they must be certified to be free of critical defects. A major defect is allowed in a program so long as the testing of a program is not affected.

2.6.1 Entry Criteria

Integration testing may only begin when:

- Unit-testing for all modules has been successful
- The sample integration application has been completed

2.6.2 Implementation

Because the bottom-up approach allows for a higher level of parallelism, easier test specification and easier product control, it has been chosen over the top-down approach and the sandwich approach. The general process to conduct bottom-up testing is:

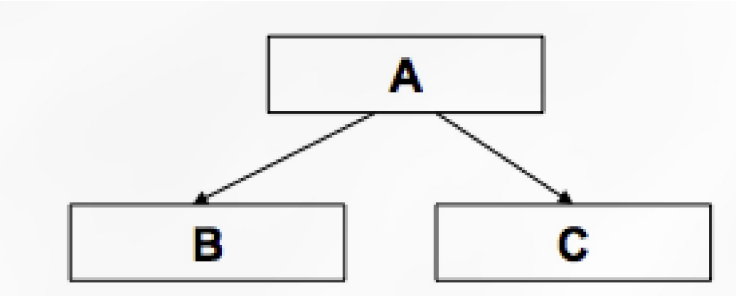

Figure 1: *Test Units A, B, C*

- Using drivers, test B and C separately.
- Create a test case for A that calls B. This way, any resulting error will be either due to A or the interface between A and B.
- Create a test case for A that calls C. This way, any resulting error will be either due to A or the interface between A and C.
- Being the most important, top-level components will be tested last.

2.7 Item Pass/Fail Criteria

Every test case must include a pass criteria in its description. In general, the two key issues encountered in software defect testing are described in the following:

- **Severity:** The extent to which the defect can impact the software. In other words, it defines the impact of a defect on the software system and can be of five varieties:
  - **Critical**: Such defects cause termination of one or more components of the system, or lead to data corruption. This renders the system unusable and there are no alternative methods to achieve the goals of the system.
  - **Major:** These defects may result in termination or data corruption, but differ from critical defects in the sense that there exists an acceptable alternative method to achieve the required results.

- **Moderate:** The defect that does not result in the termination of the system but causes the system to produce incorrect, incomplete or inconsistent results.
- **Minor:** The defect that does not result in the termination of the system nor the usability of the system, and the desired results can be easily obtained by working around the defects.
- **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and feel of the application.

- **Priority:** The order in which we should resolve a defect.
  - **Low:** The defect is an irritant which should be repaired, but the repair can be deferred until more serious defect has been fixed.
  - **Medium:** The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
  - **High:** The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been performed.

Table 10: *Defect Priority-Severity Categories*

Combination Factor	Frequency	Pass / Fail
Low Priority & Low Severity	Less than 15%	Pass
High Priority & Low Severity	Less than 5%	Pass
High Severity & Low Priority	-	Fail
High Priority & High Severity	-	Fail

2.8 Suspension Criteria and Resumption Requirements

System Integration Testing in the integration environment may be suspended under the following set of criteria:

- When the system requirements are executed adequately through test cases
- Bug reporting rate reaches a maximum threshold

- The test environment for conducting the testing no longer exists
- The scheduled time for testing is over
- The budget allocation for testing is over

System Integration Testing in the integration environment may be resumed under the following circumstances:

- When the external dependent systems become available again
- When a fix is successfully implemented and the team is notified to continue testing
- The contract is negotiated again with the client to extend delivery
- The vacation period ends

2.9 Test Deliverables

Integration testing requires the following prerequisites:

- Architectural Design Document completed
- Detailed Design Document completed
- Unit Test Plan completed

specific integration test only can begin when the following items are delivered:

- Unit testing test reports concerning the components involved (i.e. these components have been unit tested)
- Drivers for this specific integration test
- Input data
- Integration test report
- Integration test output data
- Problem reports

2.10 Environmental Needs

2.10.1 Hardware

Testing Environment:

- Server
- User web browser

2.10.2 Software

Test Case Management Tool:

- SVN

2.11 Responsibilities

Table 11: *Integration Test Roles and Responsibilities*

Role	Number of Manpower Required	Comments or Specific Responsibilities
QA Manager & Engineer	1	Defines the specific tests to be conducted. Identify and define the technical approach to the implementation of the test effort. Responsibilities include: • Recognize test ideas • Define test details • Determine test results • Document change requests • Evaluate product quality • Define test approach • Define the architecture of the test automation • Verify test techniques • Define testability elements

		• Structure test implementation
Back-End Developer	1	Ensures well maintenance of test data/database environment and assets. Responsibilities include: • Support the administration of test data and database.
Front-End Developer	1	Define and identify the operations, attributes, and associations of the test classes. Responsibilities include: • Defines the test classes required to support testability requirements as defined by the test team

2.12 Risks and Contingencies

Table 12: *Integration Test Risks and Contingencies*

Risk #	Risk	Mitigation Strategy	Probability	Severity
1	A few unimplemented features are not testable.	Identify the features to be tested and record those that will not be tested.	High	Low
2	The delay in the implementation phase, results in delay in test items.	Monitor and control the delay in critical implementation activities. If needed, amend the release scope to leave enough time for testing. And identify the critical activities during upstream planning.	High	High
3	Lack of time left for test phase.	Reschedule testing tasks. Involve the Project Manager, QA Manager and Engineer.	Medium	High
4	Delays in the fixing of identified bugs, especially critical ones.	Establish standard procedures for bug fixing and handling. Estimation for test schedule should consider the fixing	Medium	High

		time. Automate regression testing as much as possible.		
5	Absent of testing personnel.	Reschedule testing tasks. Involve the Project Manager, QA Manager and Engineer.	Low	High
6	Breakdown of the testing software or hardware tools.	Frequent back up of test cases and test results. Bug report should be stored in multiple repositories and backed up frequently. Fix the tools and reschedule the testing tasks.	Low	High
7	Impossible to cover all Android models in the market.	Test representative models that have high market shares and are expected to high demand for an amount of time.	High	Medium

2.13 Approvals

Project Manager must approve the initial test plan. When tests are successfully completed, the testers will sign off the use case requirements.