**Automated Task Management System**

Submitted by: Group 4

Submitted to:   Hao Lac

ICET Department,

School of Engineering Technology and Applied Science

Progress Campus, Block A

Centennial College

Discipline:   Software Engineering 

Due Date:  19 January 2024

# Declaration of Sole Authorship

We, Group 4, confirm that this work submitted for assessment is our own and is expressed in our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included.

Signed:

Prachi Patel – 301272750 (Software Engineering technician)

Sakshi Shingala – 301286045 (Software Engineering Technician)

Ravi – 301276475 (Software Engineering Technician)

Khushi Bishnoi – 301239973 (Software Engineering Technician)

Chamara Kotuwage Bambara – 301303823 (Software Engineering Technician)

Dilakshan Sivaruban – 301302523 (Software Engineering Technician)

Date: 19 January 2024

# Abstract

The Task Management System (TR) discussed below is an automated solution designed to transform the way assignments, tests, quizzes, and tracking are managed within organizational structures. The main goal of the project is to optimize these procedures to increase productivity overall, minimize manual errors, and boost efficiency. The system provides a user-friendly interface for easy task access by utilizing enhanced automation, guaranteeing a more transparent and organized workflow. This project is important because it will trigger a radical change in the way that organizations carry out their work. By automating repetitive operations, the suggested system can boost efficiency and lower the possibility of human error. Moreover, it claims to increase overall output through resource allocation optimization and real-time insights via extensive tracking features. It is recommended that organizations think about implementing this automated task management system as a calculated step toward achieving operational excellence. Adopting this technology innovation is a proactive approach to solving the problems related to manual work management, not just an upgrade. By putting this system into place, companies may expect more productivity, better accuracy, and a more organized method of handling tasks, all of which will help them succeed in a fast-paced work environment.**Table of Contents**

[**Declaration of Sole Authorship**](#_gjdgxs) **2**

[**Abstract**](#_30j0zll) **3**

[**List of Figures**](#_1fob9te) **6**

[**List of Tables**](#_3znysh7) **7**

[**1.0 INTRODUCTION**](#_tyjcwt) **8**

[**2.0 METHODOLOGY AND RESULTS**](#_3dy6vkm) **9**

[2.1 Literature Review](#_1t3h5sf) 9

[2.2 Proposed Solution](#_4d34og8) 9

[2.3 User Role Modelling](#_2s8eyo1) 10

[2.3.1 Brainstorm and Group](#_17dp8vu) 10

[2.3.2 Consolidated User Roles](#_26in1rg) 11

[2.3.3 Description of User Roles and Persona](#_35nkun2) 12

[2.3.4 Additional Documentation](#_1ksv4uv) 13

[2.4 Release 1.0](#_44sinio) 14

[2.4.1  User Stories](#_2jxsxqh) 14

[2.4.2  Additional Documentation](#_2bn6wsx) 18

[2.4.3  Release Plan 1.0](#_qsh70q) 19

[2.4.4  Iteration Plan (Release 1.0)](#_3as4poj) 20

[2.4.5 Additional Documentation](#_49x2ik5) 21

[2.4.6  Progress Monitoring](#_2p2csry) 22

[2.4.7  Acceptance Tests for Release 1.0](#_3o7alnk) 23

[2.5.1  User Stories](#_32hioqz) 25

[2.5.2  Additional Documentation](#_1hmsyys) 26

[2.5.3   Release Plan 2.0](#_41mghml) 27

[2.5.4   Iteration Plan (Release 2.0)](#_2grqrue) 28

[2.5.5   Additional Documentation](#_vx1227) 29

[2.5.6   Progress Monitoring](#_3fwokq0) 30

[2.5.7   Acceptance Tests for Release 2.0](#_1v1yuxt) 31

[**3.0 CONCLUSIONS**](#_4f1mdlm) **32**

[**4.0 RECOMMENDATIONS**](#_2u6wntf) **33**

[**CREDITS, LICENSE, AND REFERENCES**](#_19c6y18) **34**

[Credits](#_3tbugp1) 34

[License](#_28h4qwu) 34

[References](#_nmf14n) 34

[**APPENDIX A (DESIGN DOCUMENT)**](#_37m2jsg) **35**

[**APPENDIX B (TEST PLAN)**](#_1mrcu09) **36**

[1.0 Introduction](#_46r0co2) 36

[1.0.1   Goals](#_2lwamvv) 36

[1.0.2   Assumptions](#_111kx3o) 36

[1.0.3   Risks And Assets](#_3l18frh) 36

[Describe the elements that might positively influence testing on the project.](#_206ipza) 36

[2.0 Scope](#_4k668n3) 36

[2.0.1   Features To Be Tested](#_2zbgiuw) 36

[2.0.2    Features Not To Be Tested](#_1egqt2p) 36

[3.0 Testing Procedures](#_3ygebqi) 37

[3.0.1   Test Objectives](#_2dlolyb) 37

[3.0.2   Types Of Testing](#_sqyw64) 37

[3.0.2.1   Unit Testing](#_50zo1zu9r2nr) 37

[3.0.2.2   Integration Testing](#_xtemb8dj40mi) 37

[3.0.2.3   Acceptance Testing](#_92mli069q1ps) 37

[3.0.2.4   Stress Testing](#_aumtg8dy3dpc) 38

[3.0.2.5   Performance Testing](#_oqv2coreb80w) 38

[3.0.3   Testing Tools](#_3cqmetx) 38

[4.0 Schedule and Deliverables](#_1rvwp1q) 38

[**APPENDIX C (END-USER & ADMINISTRATOR MANUALS)**](#_4bvk7pj) **40**

# List of Figures

Figure 1: Organizing the user role cards on a table. 10

Figure 2: The consolidated role cards. 11

Figure 3: Example of a “consolidated” low-fidelity prototype. 15

Figure 4: A story with notes providing additional detail. 15

Figure 5: The revised front of a story card with only the story and questions to be discussed. 16

Figure 6: Details that imply test cases are separated from the story itself. Here they are shown on the back of the story card. 16

Figure 7: An example of a constraint story card. 17

Figure 8: Possible electronic representation of a physical story card. 17

Figure 9: Iteration burndown chart for data from Table 5. 39

# List of Tables

Table 1: The Must-Have stories for Release x.y [1]. 17

Table 2: The Should-Have stories for Release x.y [1]. 17

Table 3: Disaggregated tasks per story [1]. 20

Table 4: Stories, acceptance tests, and contributors for Release 1.0 (Green=Accepted; Red=Rejected; Black=Not started). 23

Table 5: Progress and changes for all iterations. 39

# 1.0 INTRODUCTION

The growing complexity of task management in organizations frequently results in inefficiencies, delays, and an increased risk of manual errors. To tackle these challenges, our team has embarked on the creation of an Automated Task Management System. The technical issue at hand revolves around optimizing internal processes concerning task allocation, monitoring, and fulfillment. The conventional manual approaches have been proven to be time-consuming and susceptible to mistakes, thus necessitating a technological resolution.

The goal of this project is to develop an online platform that is easy for users to navigate. It will have features like a smooth task assignment interface, real-time progress tracking, and automated notifications. The aim is to increase task completion efficiency by 20% and reduce manual errors by 15%. The system not only focuses on improving efficiency but also promotes better collaboration and communication among users. However, it's worth mentioning that the project does not include mobile applications, localized versions, or server-related aspects.

The team will develop certain completion requirements, such as a fully working website, successful user authentication, and approval through user acceptability testing, in response to the particular obstacles it faced in integrating a dependable database adapter and guaranteeing user authentication standards. This introduction emphasizes the Automated Task Management System project's potential to transform task management inside the company completely and lays the groundwork for understanding its goals, problems, and purpose.

# 2.0 METHODOLOGY AND RESULTS

## 2.1 Literature Review

The literature showcases both strengths and weaknesses that inform the development of the proposed Automated Task Management System (ATMS). Several systems have demonstrated effectiveness in addressing the challenges associated with manual task management.

Strengths:

The user-friendly interface of this task management system will make it possible to assign and monitor tasks effectively. This interface improves the user experience by making it easier for users to organize and keep track of their chores. Real-time tracking features will also prove to be quite helpful, providing transparency and the most recent data on task advancement. Automation will be a big benefit since it will help users keep aware of future chores and deadlines by sending out notifications and reminders. All of these characteristics work together to increase job completion time and workflow efficiency.

Weakness:

On the other hand, the suggested ATMS has prospects because of the noticeable shortcomings of the current systems. Some systems have excellent user interface designs, while others don't have the kind of intuitiveness that makes them widely adopted by users. This disparity in usability has the potential to impede a task management system's overall performance by provoking opposition from end users. One prevalent flaw is integration with third-party apps, which restricts the smooth transfer of data across computers. Furthermore, some systems don't have strong analytics and reporting features, which prevents businesses from learning important lessons about how best to assign resources and complete tasks.

## 2.2 Proposed Solution

The suggested solution is an Automated Task Management System created to streamline and enhance task assignment, tracking, and overall productivity within an organization. The system provides a user-friendly interface for task assignment, real-time tracking, and automated notifications. The main advantages of this solution are its capacity to greatly improve task completion efficiency, decrease the chance of manual errors, and enhance communication among users.

Strengths:

1. Efficiency Boost: The system aims to increase task completion efficiency by 20%, allowing for a more streamlined workflow and quicker access to assignments.

2. Error Reduction: Automation reduces the possibility of human errors in task assignment and tracking, contributing to a 15% reduction in manual errors.

3. Improved Communication: The centralized platform facilitates better communication and collaboration among stakeholders, ensuring everyone stays informed about task progress.

4. Real-time Tracking: Users can track the progress of tasks in real-time, providing transparency and up-to-date information on task status.

Automated Notifications: The system sends automated notifications and reminders, helping users stay on top of their tasks and deadlines.

Weakness:

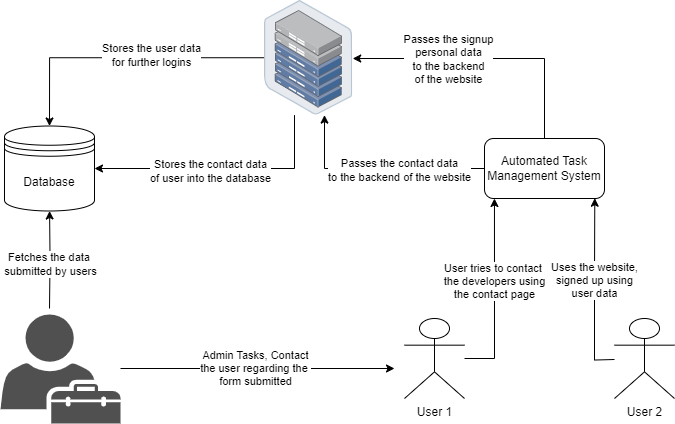
1. Scope Limitations: The project scope excludes mobile applications, third-party integrations, and localized versions, limiting the system's versatility.

2. Server Dependency: The acquisition or setup of servers for hosting the system is not within the project scope, potentially requiring additional resources for server management.

3. Limited Customization: The system may lack certain custom features beyond the defined scope, limiting adaptability to specific organizational needs.

Developer Perspective Diagram:

This diagram illustrates Automated task management infrastructure, comprising server for data operations, Database for storing user’s profiles and learning data, and users facing front-end interface. The servers interact with the databases to deliver personalised content to Users throng appropriate route, creating a dynamic and automated system.



## 2.3 User Role Modelling

### 2.3.1 Brainstorm and Group

During the initial phase, our team brainstormed and identified a range of users and their roles based on various factors, including frequency of use, expertise level, proficiency with software and application, and their goals for using the online platform. The following initial roles were identified:

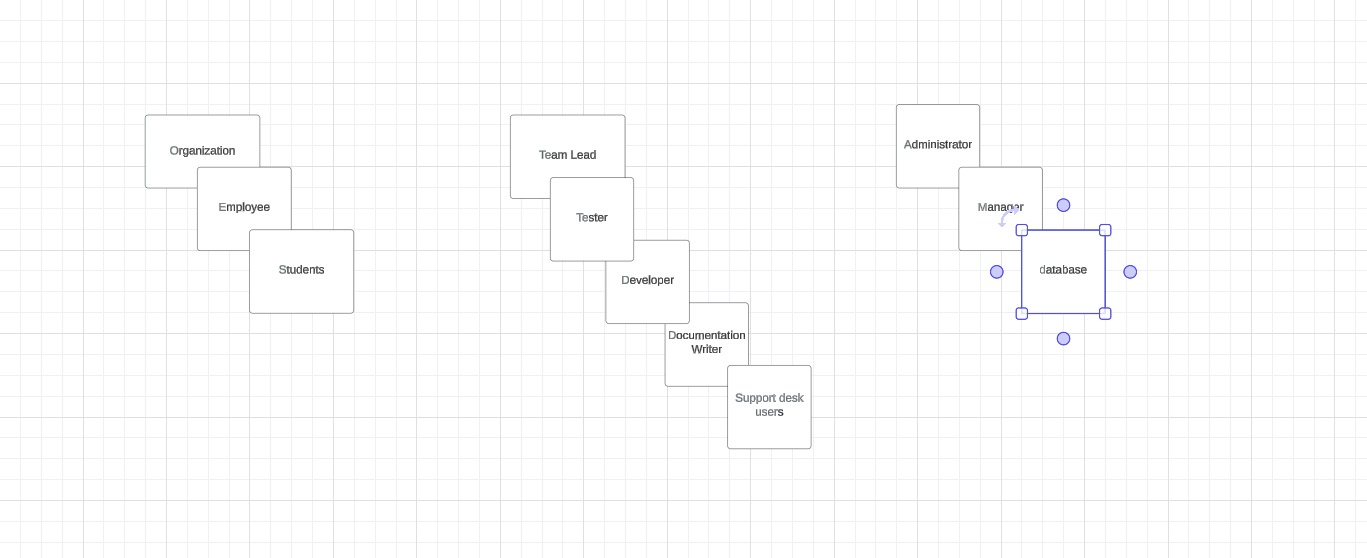


Figure 1: Organizing the user role cards in a group.

**2.3.2 Consolidated User Roles**

The initial set of user roles was divided into clusters based on resemblance criteria. The organisation process included assessing aspects such as skill level, app competence, and specific aims for utilising Automated system. Our team worked together to ensure that the organisation was logical and efficient. The end outcome was the formation of three major clusters:

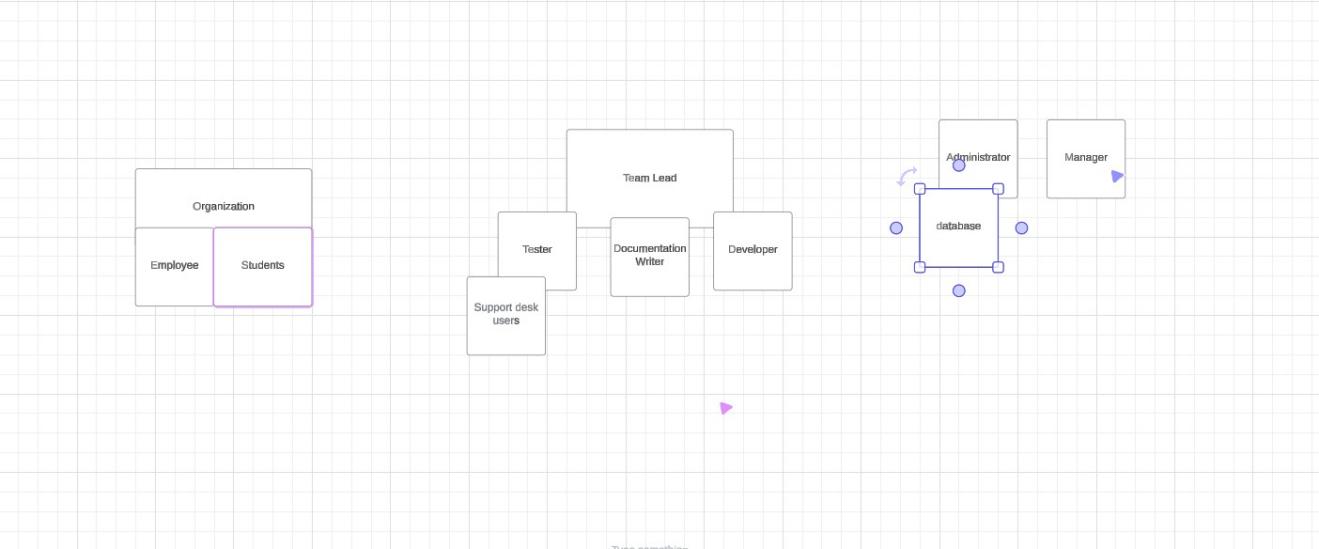


Figure 2: The consolidated user cards.

**Clusters**

1. Organization Cluster: This cluster includes the Organization, Employee, and Students roles. These roles are directly involved in the day-to-day operations and tasks within the system.

2. Team Lead Cluster: This cluster includes the Team Lead, Tester, Developer, and Documentation Writer roles. These roles are more specialized and have specific responsibilities within the system.

3. Administrator Cluster: This cluster includes the Administrator and Manager roles. These roles have the highest level of access and control in the system. They are responsible for strategic decision-making and system management.

**2.3.3 User Roles and Clusters Analysis**

**User Roles**

1. Organization: This role represents the overarching entity within which the system operates. It could be a company, a department, or a team. The Organization sets the overall goals and objectives that the system aims to achieve.

2. Employee: Employees are the primary users of the system. They are responsible for executing tasks and assignments within the system. Their roles may vary, but they all contribute to the overall objectives of the Organization.

3. Students: In the context of an educational organization, students are also considered users of the system. They are responsible for completing assignments and tasks assigned to them within the system.

4. Team Lead: The Team Lead oversees a group of Employees or Students. They are responsible for assigning tasks, tracking progress, and ensuring that their specific team's objectives align with those of the Organization.

5. Tester: The Tester is responsible for ensuring the system functions as expected. They conduct various tests to identify and fix bugs and errors.

6. Developer: The Developer is responsible for the technical aspects of the system. They design, build, and maintain the system's functionalities.

7. Documentation Writer: The Documentation Writer is responsible for creating user manuals and technical documentation to support the system's ongoing use and maintenance.

8. Support Desk Users: Support Desk Users provide assistance and support to other users of the system. They help resolve issues and answer queries about the system.

9. Administrator: The Administrator has the highest level of access in the system. They ensure the smooth operation and configuration of the Automated Task Management System (ATMS).

10. Manager: The Manager oversees the entire operation. They make strategic decisions and ensure that all user roles work together towards the Organization's objectives.

### 2.3.4 Additional Documentation

YouTube Link: - <https://youtu.be/SHaaC1gisB0>

Name: COMP 231| User Role Modelling

## 2.4 Release 1.0

### 2.4.1 User Stories

The following are required for this section:

1. Show and discuss the results of your low-fidelity prototype generated during your story-writing workshop (a sample of a “consolidated” low-fidelity prototype is illustrated in Figure 3).
2. Provide your definition of story point.
3. Show the stories created during the story-writing workshop.  You can submit scanned images of your index cards (both front and back). Figures 4 to 7 illustrates a single story with variation on the *Note*s (Figures 4 and 5), acceptance tests shown on the back of the index card (Figure 6), and a constraint, or non-functional requirement (Figure 7).
4. Prioritized stories based on the MoSCoW rule as illustrated in Tables 1 and 2 (see also *User Stories* deliverable).

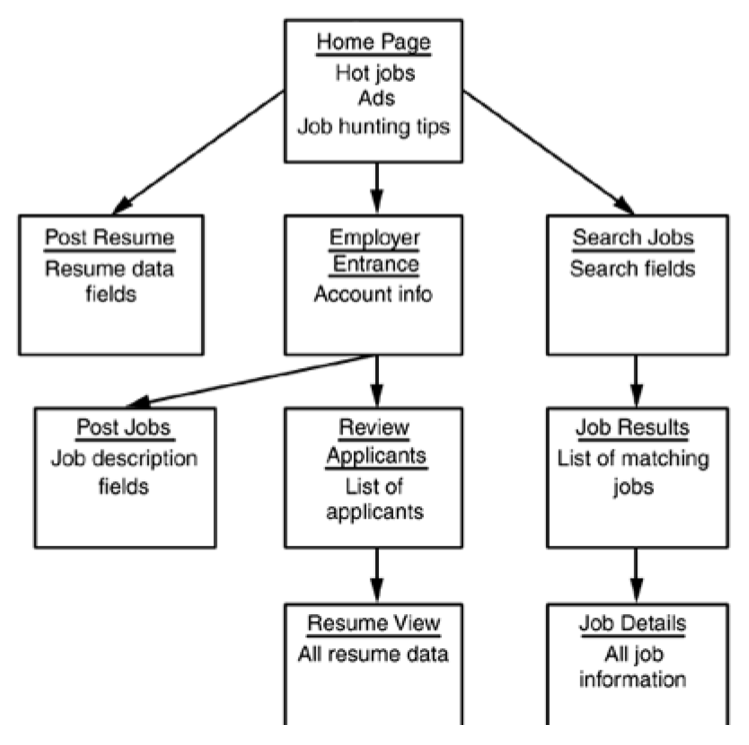


Figure 3: Example of a “consolidated” low-fidelity prototype. Note that each “individual” low-fidelity prototype is developed for each user role [1].

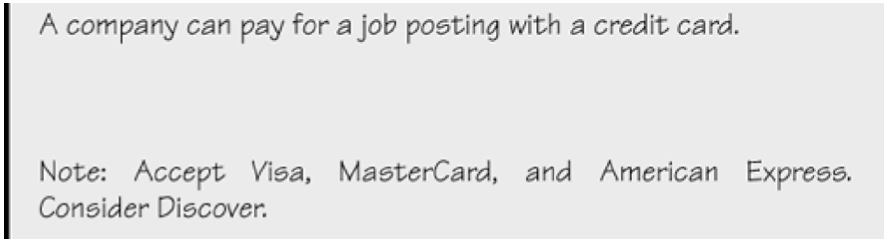


Figure 4: A story with notes providing additional detail [1].

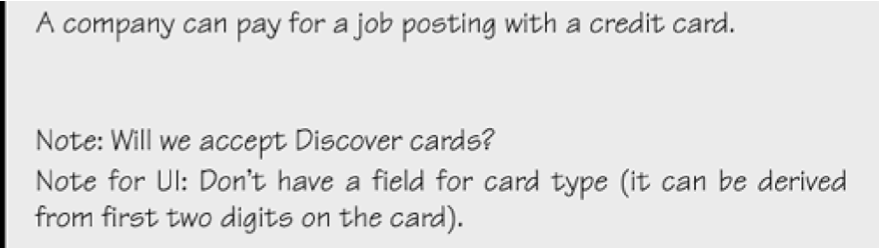


Figure 5: The revised front of a story card with only the story and questions to be discussed [1].



Figure 6: Details that imply test cases are separated from the story itself. Here they are shown on the back of the story card [1].

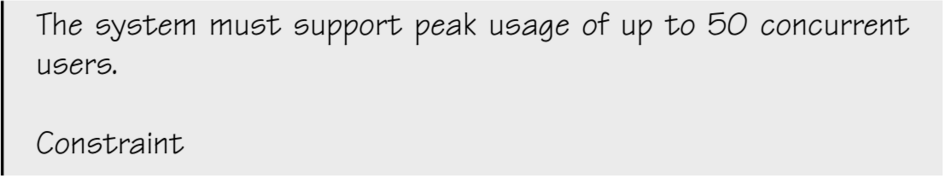


Figure 7: An example of a constraint story card [1].

Figure 8 illustrates a possible electronic representation of a physical story card. The left column represents the front of the card while the right column represents the back of the card.

|  |  |
| --- | --- |
| A Company can pay for a job posting with a credit card.  Note: Will we accept Discover cards?  Note for UI: Don’t have a field for card type (it can be derived from the first two digits on the card)  Estimate: 3 hrs. | Test with Visa, MasterCard and American Express.  Expected outcome: the system should automatically display a label of the card type.  Test with Diner’s Club.  Expected outcome: the system should prompt the user for a Visa, MasterCard or American Express card.  ...*<rest of the Tests follows>* |

Figure 8: Possible electronic representation of a physical story card.

Table 1: The Must-Have stories for Release x.y [1].

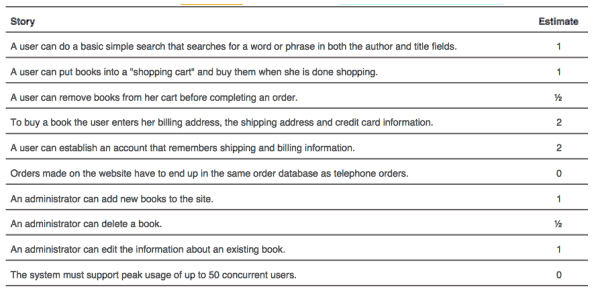
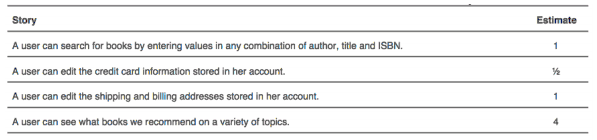


Table 2: The Should-Have stories for Release x.y [1].



### 2.4.2 Additional Documentation

For this section, include the video(s) from your workshop showing how your team:

1. Brainstormed for stories and generated the low-fidelity prototype (story writing workshop).
2. Estimated stories using the Wideband Delphi approach.
3. Prioritized stories using the MoSCoW rule.

Provide the file name and URL to the video(s) in your shared folder or YouTube channel.

### 2.4.3 Release Plan 1.0

The following are required for this section[[1]](#footnote-1):

1. Provide the product development roadmap.
2. Provide the iteration length and the release date.
3. The refine priorities of the Must- and Should-Have stories by organizing the stories into groups that have a high likelihood of being performed together.
4. The actual release plan.
5. Place the contents of your paper prototype in [Appendix A (Design Document)](#qoacug2ru8z).

### 2.4.4 Iteration Plan (Release 1.0)

The following are required for this section:

1. Present each iteration plan with tables showing disaggregated tasks per story; a sample is shown in Table 3. See also the *Planning an Iteration* deliverable.
2. Discuss any discrepancies between the estimated and actual ideal time required to complete the tasks for the Table mentioned above.

|  |
| --- |
| Table 3: Disaggregated tasks per story [1].  https://lh4.googleusercontent.com/XeQi1HulutY2JRQ6keiIBwQAABwmYtP7t1GjQFo1b4WaGRlCZDzp_VFe0oAvqmD85w5JDufu7dIFrP2Z7WLEBL2hjhkyLrpqtic6cLaESTPtdGqlVWXe6H9yRPLc_mYB_TqyvmU |

### 2.4.5 Additional Documentation

For this section, include 1 of 4 videos from your Iteration Planning meetings (recall that you have a total of 4 Iteration Planning meetings)[[2]](#footnote-2):

1. Showing how your team disaggregated stories into their constituent tasks.
2. How developers on your team volunteer and take responsibilities for tasks.

Provide the file name and URL to the video(s) in your shared folder or YouTube channel.

### 

### 2.4.7 Acceptance Tests for Release 1.0

The following are required for this section:

1. A table of stories and their associated acceptance tests for this Release as shown below in the sample in Table 5.
2. The link to your video demo for Release 1.0 stored either in a cloud drive, or your YouTube channel.

Table 4: Stories, acceptance tests, and contributors for Release 1.0 (Green=Passed; Red=Failed).

|  |  |  |
| --- | --- | --- |
| **Full description of user story** | **Acceptance test(s)** | **Name(s) of contributing Developer(s)** |
| As an User, I can … so that ….[[3]](#footnote-3) | Test with inputs ….  Expected outcome: ...  Test with inputs ….  Expected outcome: ... | Susan Smith,  Jay Johnson |
| As an Administrator, I can … so that ….[[4]](#footnote-4) | Test with inputs ….  Expected outcome: ...  Test with inputs ….  Expected outcome: ...  Test with inputs ….  Expected outcome: ... | Susan Smith,  Jay Johnson,  Shannon Shore,  George Gavinson |
| As an User, I can … so that …. | Test with inputs ….  Expected outcome: ...  Test with inputs ….  Expected outcome: ...  Test with inputs ….  Expected outcome: ... | Jay Johnson,  Shannon Shore,  George Gavinson |
| As an User, I can … so that ….[[5]](#footnote-5) | Test with inputs ….  Expected outcome: ... | Shannon Shore |
| As a Guest, I can … so that …. | Test with inputs ….  Expected outcome: ...  Test with inputs ….  Expected outcome: ...  Test with inputs ….  Expected outcome: ... | Susan Smith,  Jay Johnson,  Shannon Shore,  George Gavinson,  Abbey Appleby,  Brian Bolt |

*<Insert url to video demo of Release 1.0 here>*

## 2.5 Release 2.0

Release 2.0 has essentially the same structure as Release 1.0.

### 2.5.1 User Stories

If your team wrote enough stories to cover up to or beyond Release 2.0 during your first story-writing workshop as described in the *User Stories* section 2.4.1, then your team will not need to hold a second formal workshop.

If a second workshop was held, submission for this section is the same as section 2.4.1.

### 2.5.2 Additional Documentation

Include this section in your Technical Report only if your team required a second formal story-writing workshop. If a second workshop was held, submission for this section is the same as section 2.4.2.

### 2.5.3   Release Plan 2.0

The requirements for this section are the same as section 2.4.3; update or add sections if required.

### 2.5.4   Iteration Plan (Release 2.0)

The requirements for this section are the same as section 2.4.4.

### 2.5.5   Additional Documentation

This section is required ONLY IF your team submitted materials for section 2.4.5.

### 2.5.7   Acceptance Tests for Release 2.0

The requirements for this section follow the same requirements as in section 2.4.7 except acceptance testing is for stories allocated for Release 2.0 and incomplete stories subsequently moved from Release 1.0.

# 3.0 CONCLUSIONS

A conclusion interprets the data found in the Body. It is reasoned judgment and not opinions. Consider the variables. Relate cause and effect. Analyze, evaluate, make comparisons and contrasts. Base the conclusion on fact.

# 4.0 RECOMMENDATIONS

Recommendations are not required for all studies. They suggest a course of action and would generally be provided when there are additional areas for study, or if the reason for the TR was to determine the best action going forward.

# CREDITS, LICENSE, AND REFERENCES

## Credits

Provide any credits here. The following are examples:

Author of the template graphic layout : Hao Lac <haolac.at.centennial@gmail.com>

Author of the template explanation text : John Doe <[john.doe@centennialcollege.ca](mailto:john.doe@centennialcollege.ca)>

## License

State the license granted with your system. For example:

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the appendix entitled "GNU Free Documentation License".

## References

[1] Cohn, Mike. 2004. *User Stories Applied: For Agile Software Development*, Addison-Wesley Professional.

# APPENDIX A (DESIGN DOCUMENT)

Traditional approaches to software development, in contrast to that of Agile approaches, place a great deal of emphasis on upfront design. The Agile approach to design is quick sessions that seek the simplest solution and then incrementally build on that solution. A quick design session can include the use of CRC cards that can ultimately lead to the generation of UML diagrams.

Using Agile approaches to software development does not mean you are limited to using only Agile techniques. If you feel that a technique (e.g., use case or interaction design scenario) is more suitable, or better conveys the features of your system to your users, then use it.

In this section, you are required to submit and discuss the following:

* A paper prototype of your application/system.
* Any design work your team has done in developing your system including CRC cards, UML diagrams, ERD diagrams, use cases, interaction design scenario, etc.

# APPENDIX B (TEST PLAN)

## 1.0 Introduction

### 1.0.1   Goals

Summarize the testing goals for the project.

### 1.0.2   Assumptions

Any assumptions which may affect the understanding or execution of this plan should be recorded here.

### 1.0.3   Risks And Assets

Describe the elements (software or hardware) that are not part of your application but still may impact its correctness and must be checked.

### Describe the elements that might positively influence testing on the project.

## 2.0 Scope

### 2.0.1   Features To Be Tested

Describe the features and functions that will be tested during the project. This should include functional and non-functional requirements.

### 2.0.2    Features Not To Be Tested

Describe the features that will not be tested and reason why.

## 3.0 Testing Procedures

Describe the testing procedures that the project will use. This includes the test lifecycle, types of testing, test objectives, and test criteria.

### 3.0.1   Test Objectives

Describe the objectives of the testing process.

### 3.0.2   Types Of Testing

Describe the types of testing that the project will use.

#### 3.0.2.1   Unit Testing

Describe the strategy for unit testing of the individual subsystems. This includes an indication of the subsystems that will undergo unit tests or the criteria to be used to select subsystems for unit test. Test cases are NOT included here.

#### 3.0.2.2   Integration Testing

Specify the integration testing strategy used. Describe the tests that will be performed in order to verify the interfaces between the subsystems of the software system. This section includes a discussion of the order of integration of subsystems. Test cases are NOT included here.

#### 3.0.2.3   Acceptance Testing

Specify the strategy for testing the software once it has been deployed. This section includes a discussion of the order of acceptance by software function. Test cases are NOT included here.

#### 3.0.2.4   Stress Testing

Identify the limits under which the program is expected to perform (memory constraints, disk space constraints, etc).

#### 3.0.2.5   Performance Testing

Refer to the functional requirements that specify acceptable performance.

### 3.0.3   Testing Tools

Describe the tools that you will use for testing.

## 4.0 Schedule and Deliverables

Describe the test deliverables that will be created during the project lifecycle. Include two tables, one for the schedule of tasks, another for the list of deliverables:

* Acceptance test
* Unit test
* System/Integration test
* Stress test
* Performance test
* Screen prototypes
* Defect reports and summaries
* Test logs and reports

Describe the reports that will be generated by the testing process.

Examples include:

Test Summary Report - A final report of the testing results from the project. Can include items such as total number of test cases, number of test cases executed, % test cases passed, etc.

# APPENDIX C (END-USER & ADMINISTRATOR MANUALS)

In this section, include a user manual for your system/application. The user manual should include the following items:

1. Instructions on how to install and configure your system/application, documenting all external software dependencies that need to be setup manually.
2. A user guide for the administrator (use screen shots of your system/application and briefly discuss each screen shot).
3. A user guide for the normal user (use screen shots of your system/application and briefly discuss each screen shot).

# APPENDIX D (PROGRESS MONITORING)

Your team is required to report two items related to progress monitoring in this appendix. The first item is a table summarizing progress and changes during a release with supporting discussion; a sample is shown in [Table 5](#v6f31rop6dwk). Notice in Table 5 that all iterations are shown per Release[[6]](#footnote-6). Also, see *Table 1* in the *Measuring and Monitoring Progress* deliverable.

|  |  |  |
| --- | --- | --- |
| Table 5: Progress and changes for all iterations [1].  https://lh5.googleusercontent.com/4Ap6uEsBxYifEjZuDn1Lj7V4URVZyBH7pdlR3GI1Muc-OxP7iE51R_qXc06cBoPtAJLqXna8S1RwR7DO1t-PTcJj0Jv5ybINsjn94Z9SUdpIpFBFQnJxG7flKVvm15qJHJB4H9Y  The second item is an iteration burndown chart (see [Figure 9](#uotmwo80zdj)) reflecting the data from Table 5.   |  | | --- | |  | | Figure 9: Iteration burndown chart for data from [Table 5](#v6f31rop6dwk). | |

1. See *The Release Plan* deliverable. [↑](#footnote-ref-1)
2. Indicate which iteration the video corresponds to. If you decide to submit a video in Release 1.0, then you do not need to include an *Additional Documentation* section for Release 2.0. [↑](#footnote-ref-2)
3. Green colour code indicates that all tests passed successfully as intended. [↑](#footnote-ref-3)
4. Red colour code indicates that at least one test unintendedly failed. [↑](#footnote-ref-4)
5. When all tests for a given story fails, this may suggest that implementation of the story has not even begun and indicates poor planning on the part of the team. [↑](#footnote-ref-5)
6. For subsequent Releases, do NOT restart numbering the Iteration.  For example, let us assume that we have another Release (i.e., Release 2.0), we would continue numbering our Iterations as *Iteration 5, Iteration 6,* and so on. [↑](#footnote-ref-6)