MerchantDice

[Project Plan]
Version 1.5

By: Andre Lim, Andre Lee, Andrew Ng, Lim Kaisheng, Ling Yin, Trevor Lim, Yang Yang

Revision History

Revision Number	Date	Primary Authors(s)	Comments
1.0	22 Sep 2022	Ling Yin	First Version
1.1	29 Sep 2022	Andre Lee	Added Introduction, Project Organization and Process Definition
1.2	2 Oct 2022	Andre Lee	Added Project Estimates
1.3	8 Oct 2022	Andre Lee	Added Product Checklist, Best Practices Checklist, Risk Management
1.4	9 Oct 2022	Andre Lee	Added Gantt Chart, Work Breakdown Structure, Work Packages, Quality Assurance, Monitor and Control
1.5	12 Oct 2022	Ling Yin	Added Activity Dependencies, Work package details Minor edits to phrasing in certain sections

Table of Contents

1 Introduction

1.1 Project Overview

MerchantDice is a web application that aims to facilitate the process of thrift shopping. In doing so, buyers and sellers can come together to exchange goods conveniently via the online space. Buyers can use the platform to look for goods that suit their needs and wants easily at a low price. And sellers can use the platform to sell pre-loved items. In addition, there is a chat function that enables all users to communicate with each other. This can be used to discuss details about the transaction or for any other purposes necessary.

1.2 Project Description and Scope

MerchantDice is a one stop platform that allows thrift shoppers and sellers to exchange goods. It is aimed to improve the thrifting experience and to improve the ease of thrifting in Singapore. At the same time, as a society, we will be able to enjoy the environmental benefits of thrifting and mitigate the environmental impacts of quick-moving trends such as fast fashion.

MerchantDice is a web application which has the following functionalities. Firstly, sellers are able to upload images of the items they are selling, exercise control in pricing their items and name their items. After sellers upload their pre-loved items to the site, potential buyers are able to view the listing. They will be able to see pictures of the item, the price (specified by the seller) and the seller rating. The seller rating will indicate how previous buyers felt about their transactions with this particular seller. If the buyer has questions about the product he or she is about to purchase, they will be able to make use of the chatroom function to communicate with the seller.

After completing the transaction, the buyers will be able to rate their experience with the seller. This will allow other users to know the reliability of the seller for future transactions.

Other than that, users are also able to make use of a “favorite” function to like items. They can view items that they have liked anytime. They will be able to view their purchase history to see what items they have bought or sold before.

The design objectives are listed as:

(1) For sellers, to be able to sell their pre loved items
(2) For buyers, to be able to search for specific items
(3) For buyers, to be able to favorite specific items
(4) For buyers, to be able to buy for specific items
(5) For buyers and sellers to be able to communicate with each other via the platform
(6) For buyers, to be able to rate their transactions with a specific seller.

2 Project Organization

2.1 Team Structure

The following is the list of executive roles, as required by CMM level 3.

- Project Manager: Ling Yin
- Lead Developer: Andre Lee
- Frontend Developer: Andre Lim
- Backend Developer: Trevor Lim
- QA Manager: Kai Sheng
- QA Engineer: Andrew Ng
- Release Manager: Yang Yang

2.2 Roles and Responsibilities

Project Manager: Ling Yin

- Oversees project progress
- Approves and executes project plan
- Assigns tasks and reports status of project to team members
- Manages and motivates team members
- Represents the team to the outside world

Lead Developer: Andre Lee

- Overall Technical Lead
- Responsible for Product Release
- Coordinate software development in the team
- Ensure effective communication between team members

Frontend Developer: Andre Lim

- Implement the visual elements of the final product
- Ensure the technical feasibility of the UI/UX designs
- Assure all user input is validated before sending it to the backend

Backend Developer: Trevor Lim

- Responsible for server-side application logic
- Work closely with Frontend during integration
- Database design and implementation

Quality Assurance Manager: Kai Sheng

- Oversee the overall product and process quality
- Recording, analyzing and distributing statistical information
- Supervising QA engineer

Quality Assurance Engineer: Andrew Ng

- Ensures acceptable software quality
- Designs testing strategies
- Creates and manages test plan

- Verify software requirements
- Executes test procedure

Release Manager: Yang Yang

- Create baselines, build and integrate changes for delivery
- Manage release of product
- Maintains and monitors software builds

2.3 Team Communication

Team APT200 communication channels include the following:

- Weekly online meeting via Zoom
- Telegram group chat
- Collaborative workspace on Jira

3 Process Definition

3.1 Lifecycle Model

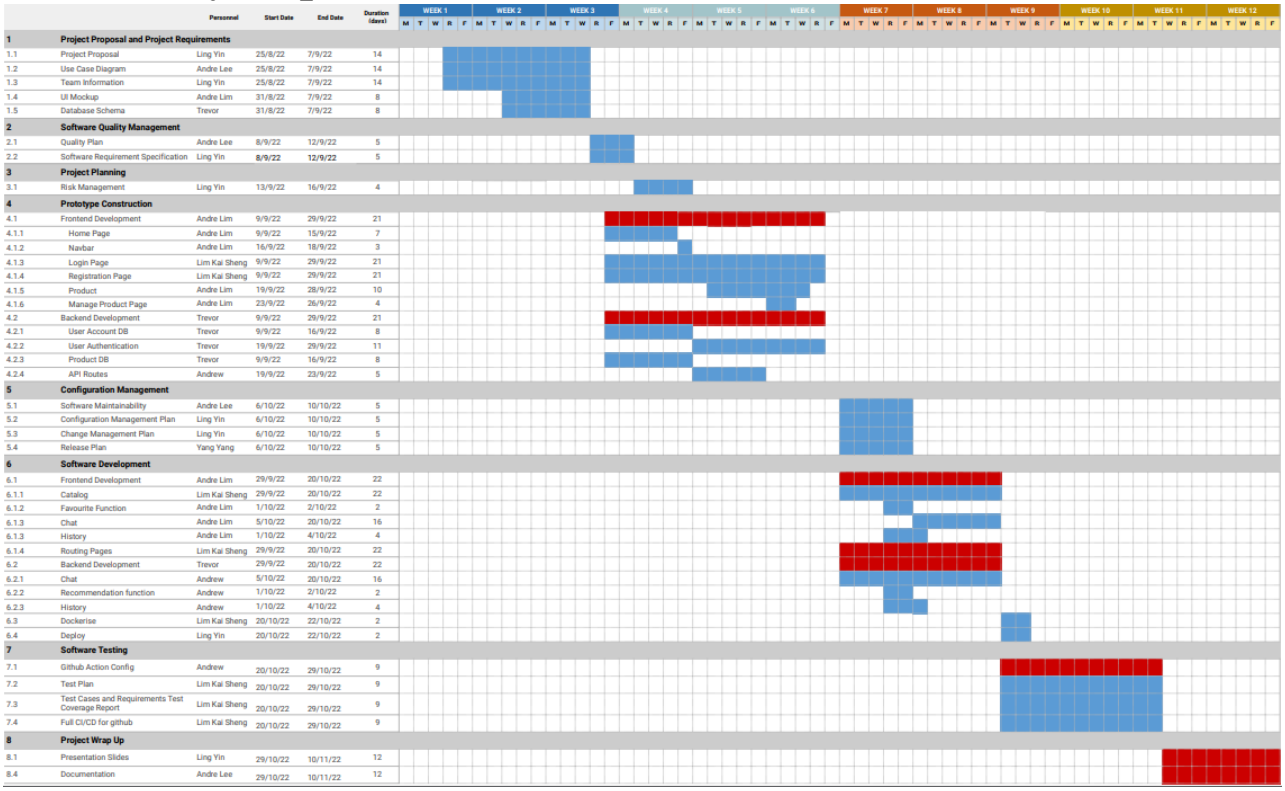
Team APT200 intends to use the Incremental Development Model throughout the MerchantDice project time span. This methodology is more flexible than the traditional Waterfall SDLC due to repeated iterations involving design, coding, unit testing, integration, and quality assurance. The Waterfall SDLC is not a viable choice due to the short timeline available for the MerchantDice project to reach delivery quality.

APT200 has chosen to avoid methodologies such as the Spiral model due to concerns over the short timeline. Should design procedures need to be revisited within the first release date, it is likely that the project will overshoot its critical schedule.

APT200 intends to deliver the first iteration of functionality on the System Delivery date indicated in the Estimations section of this document. After further client interaction, further iterations may occur as necessary.

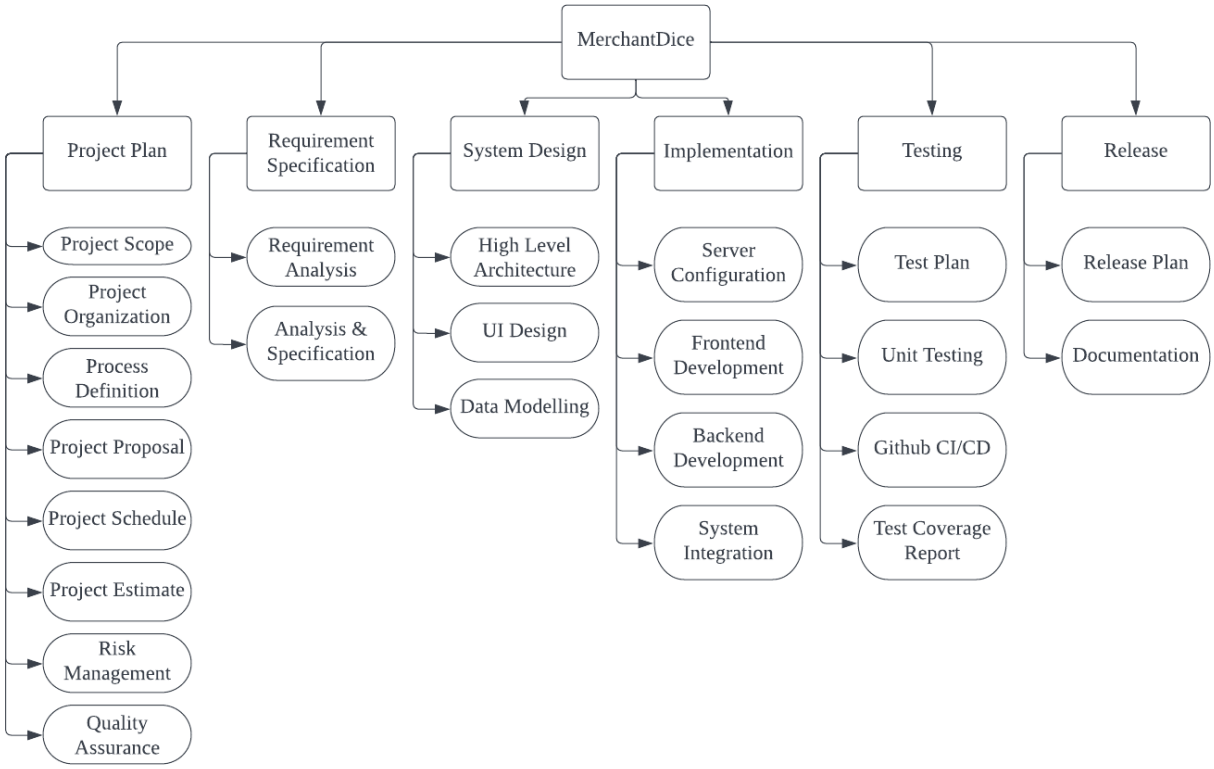
4 Schedule

4.1 Activity Dependencies and Schedule



		Personnel	Start Date	End Date	Duration (days)
1	**Project Proposal and Project Requirements**				
1.1	Project Proposal	Ling Yin	25/8/22	7/9/22	14
1.2	Use Case Diagram	Andre Lee	25/8/22	7/9/22	14
1.3	Team Information	Ling Yin	25/8/22	7/9/22	14
1.4	UI Mockup	Andre Lim	31/8/22	7/9/22	8
1.5	Database Schema	Trevor	31/8/22	7/9/22	8
2	**Software Quality Management**				
2.1	Quality Plan	Andre Lee	8/9/22	12/9/22	5
2.2	Software Requirement Specification	Ling Yin	**8/9/22**	**12/9/22**	5
3	**Project Planning**				
3.1	Risk Management	Ling Yin	13/9/22	16/9/22	4
4	**Prototype Construction**				
4.1	Frontend Development	Andre Lim	9/9/22	29/9/22	21
4.1.1	Home Page	Andre Lim	9/9/22	15/9/22	7
4.1.2	Navbar	Andre Lim	16/9/22	18/9/22	3
4.1.3	Login Page	Lim Kai Sheng	9/9/22	29/9/22	21
4.1.4	Registration Page	Lim Kai Sheng	9/9/22	29/9/22	21
4.1.5	Product	Andre Lim	19/9/22	28/9/22	10
4.1.6	Manage Product Page	Andre Lim	23/9/22	26/9/22	4
4.2	Backend Development	Trevor	9/9/22	29/9/22	21
4.2.1	User Account DB	Trevor	9/9/22	16/9/22	8
4.2.2	User Authentication	Trevor	19/9/22	29/9/22	11
4.2.3	Product DB	Trevor	9/9/22	16/9/22	8
4.2.4	API Routes	Andrew	19/9/22	23/9/22	5
5	**Configuration Management**				
5.1	Software Maintainability	Andre Lee	6/10/22	10/10/22	5
5.2	Configuration Management Plan	Ling Yin	6/10/22	10/10/22	5
5.3	Change Management Plan	Ling Yin	6/10/22	10/10/22	5
5.4	Release Plan	Yang Yang	6/10/22	10/10/22	5
6	**Software Development**				
6.1	Frontend Development	Andre Lim	29/9/22	20/10/22	22
6.1.1	Catalog	Lim Kai Sheng	29/9/22	20/10/22	22
6.1.2	Favourite Function	Andre Lim	1/10/22	2/10/22	2
6.1.3	Chat	Andre Lim	5/10/22	20/10/22	16
6.1.3	History	Andre Lim	1/10/22	4/10/22	4
6.1.4	Routing Pages	Lim Kai Sheng	29/9/22	20/10/22	22
6.2	Backend Development	Trevor	29/9/22	20/10/22	22
6.2.1	Chat	Andrew	5/10/22	20/10/22	16
6.2.2	Recommendation function	Andrew	1/10/22	2/10/22	2
6.2.3	History	Andrew	1/10/22	4/10/22	4
6.3	Dockerise	Lim Kai Sheng	20/10/22	22/10/22	2
6.4	Deploy	Ling Yin	20/10/22	22/10/22	2
7	**Software Testing**				
7.1	Github Action Config	Andrew	20/10/22	29/10/22	9
7.2	Test Plan	Lim Kai Sheng	20/10/22	29/10/22	9
7.3	Test Cases and Requirements Test Coverage Report	Lim Kai Sheng	20/10/22	29/10/22	9
7.4	Full CI/CD for github	Lim Kai Sheng	20/10/22	29/10/22	9
8	**Project Wrap Up**				
8.1	Presentation Slides	Ling Yin	29/10/22	10/11/22	12
8.4	Documentation	Andre Lee	29/10/22	10/11/22	12

4.2 Work Breakdown Structure



4.3 Work Package Details

The entire project work is broken down by the important phases of the software development life cycle. They include the following:

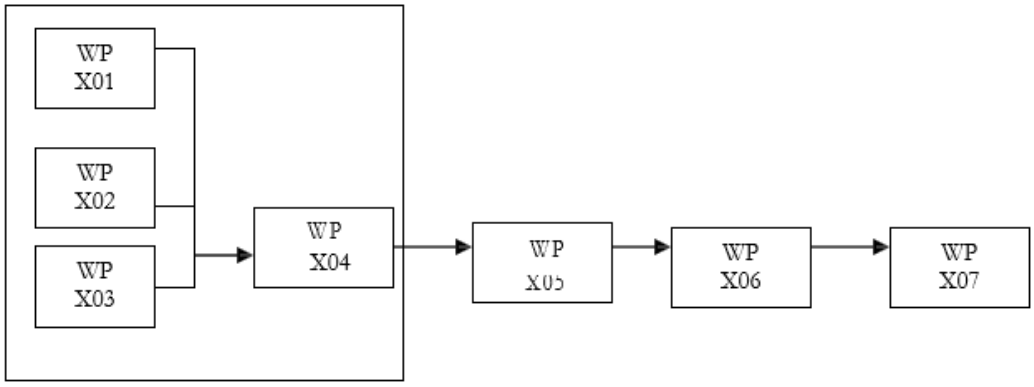
1. Project Plan
2. Requirement Specification
3. User Interface
4. Technical Architecture
5. Data Modeling
6. Coding & Unit Testing
7. Integration & Quality Assurance

4.4 Activity Dependencies

The following table describes the dependencies of the deliverable work packages:

Work Package #	Work Package Description	Duration	Dependencies
X01	Project Plan	14 days	--
X02	Requirement Specification	7 days	--
X03	User Interface	14 days	--
X04	Technical Architecture	12.1 days	X01,X02,X03
X05	Data Modeling	7 days	X04
X06	Coding & Unit Testing	16.2 days	X05
X07	Integration & System Testing	16.2 days	X06

The following Activity Network Diagram describes the above in more graphical detail:



Note that work package X05 is dependent on all work packages encapsulated by the larger boxes linked to its left. For instance, WP X05 may not start until WP X01- X04 has been finished.

4.5 Work Package Details

Work packages are listed below. A team member, indicated in bold, has been assigned as primarily responsible for each work package and will coordinate that package.

Project	MerchantDice
Work Package	X01 Project Plan
Assigned to	**Ling Yin**, Andre Lee
Effort	14 PD
Start Date	Thursday, 22 September 2022
Purpose	To determine an introductory overview of the project, to be refined in later work packages.
Inputs	None
Activities	This work package includes providing a brief overview of the project, its objectives, and a set of proposed project deliverables throughout the development of the software cycle. The people responsible for this work package will also be transcribing ideas brought up in the group meeting discussion into a formal report.
Outputs	A written document of the Project Plan Introduction

Project	MerchantDice
Work Package	X02 Requirement Specification
Assigned to	**Ling Yin**, Andre Lee, Andre Lim, Andrew Ng, Lim Kaisheng, Yang Yang, Trevor
Effort	7PD
Start Date	Thursday, 25 August 2022
Purpose	To establish a common understanding between the customer and the software project team of the customers' requirements to be addressed by the project
Inputs	Customer's requirements
Activities	Identify "the customer", interview customer, write and inspect customer requirement and build requirements.

Outputs	A written document of the requirement specification

Project	MerchantDice
Work Package	X03 User Interface
Assigned to	**Andre Lim**, Lim Kaisheng
Effort	7PD
Start Date	Thursday, 25 August 2022
Purpose	To build the user interface between the system and the customer, to make it easy use, and friendly to the customer
Inputs	User information, Product information
Activities	To get product information and display them on the screen. To get user information and display the result of the user request
Outputs	User interface

Project	MerchantDice
Work Package	X04 Technical Architecture
Assigned to	**Andre Lee**, Andre Lim. Trevor Lim
Effort	14PD
Start Date	Thursday, 25 August 2022
Purpose	To do the high level architecture design
Inputs	Project Plan Work Packages (X01 to X03 inclusive).
Activities	High level design entails defining the architecture of the software system and identifying the various components and how they are inter-related to and interactive with each other. Designers also need to decide on the software and hardware infrastructures, such as what operating system on which the software is built, the language used to implement the software, and so on. Design topics including maintainability, portability, and reusability will be addressed here as well.

Outputs	High Level Design and Architectural Specification.

Project	MerchantDice
Work Package	X05 Data Modelling
Assigned to	**Andre Lee**, Trevor Lim,
Effort	To build the project's database
Start Date	Thursday, 8 September 2022
Purpose	To build the project's database
Inputs	Project Plan Work Packages (X01 to X05 inclusive).
Activities	Analyze the data flow relationships, entity relationships
Outputs	Database set up. API routes set up

Project	MerchantDice
Work Package	X07 Integration and System Testing
Assigned to	**Lim Kaisheng,** Yang Yang, Andre Lee
Effort	16.2PD
Start Date	Thursday, 22 September 2022
Purpose	To identify and fix logical and syntactical errors produced during the implementation of the System, and setting up drivers and stubs to see how the module responds to various inputs. Black box testing as well as white box testing might be conducted to check for logical errors. All the testing procedures will be documented in the Test Plan report. If problems are found, they will be noted and fixed at the earliest possible time.
Inputs	Project Plan Work Package X07.
Activities	The Integration testing team may try to simulate how a user might interact with the system. Similar to Unit Testing, Integration Testing may require the development of stubs and drivers as well, but here this is more geared towards the higher (overall system) level. Testers may also examine issues such as system performance and integrity. Heuristics assessment plays an important role in this work package, as intelligence components will define eventual system success.

Outputs	A test report.

5 Project Estimates

5.1 Code Size Estimation using Function Points

We calculated unadjusted function point based on the complexity of functions provided by this system. Code size is then estimated by adjusted function point.

5.1.1 Unadjusted Function Points

MerchantDice supports the following proposed functions:

User:

- Create/Update/Delete user account
- Filter sorted products based on user's entered criteria
- Favorite products to the user's account
- Write/View/Update/Delete product reviews
- Chat functionality with merchants

Merchant:

- Create/Update/Delete merchant account
- Create/Update/Delete Product listing
- View product reviews
- Chat functionality with buyers

The measure of unadjusted function points is based on five primary component elements of these functions: Inputs, Outputs, Inquiries, Logical Files, and Interfaces. Each element ranges from Low Complexity, Medium Complexity to High Complexity. The detailed evaluation of the complexity is as follows:

Rating Inputs:

- Gathering user account information: (email address, name, username, password)
- Gathering product information (product name, description, price, pick-up location, category, image)
- Preferences (i.e. favorite products) and Constraints (i.e. sorting the results by product type, price, or name)

Files Type Referenced (FTR)	Data Elements		
	1-4	5-15	Greater than 15
Less than 2	Low (3)	Low (3)	Average (4)
2	Low (3)	Average (4)	High (6)
Greater than 2	Average (4)	High (6)	High (6)

Rating Outputs:

- Displaying a list of the results matching the user's search criteria (product type, price or name)
- Displaying merchant product listings
- Displaying user account information

File Types Referenced (FTR)	Data Elements		
	1-5	6-19	Greater than 19
less than 2	Low (4)	Low (4)	Average (5)
2 or 3	Low (4)	Average (5)	High (7)
Greater than 3	Average (5)	High (7)	High (7)

Rating Inquiries:

- Selecting the product according the user's criteria
- Selecting user transaction information
- Selecting user account information

File Types Referenced (FTR)	Data Elements		
	1-5	6-19	Greater than 19
less than 2	Low (3)	Low (3)	Average (4)
2 or 3	Low (3)	Average (4)	High (6)
Greater than 3	Average (4)	High (6)	High (6)

Rating Logical Files:

- Transaction Details
- User Account

Record Element Types (RET)	Data Elements		
	1 to 19	20 - 50	51 or More
1 RET	Low (7)	Low(7)	Average (10)
2 to 5 RET	Low (7)	Average (10)	High (15)
6 or More RET	Average (10)	High (15)	High (15)

Rating Interfaces:

- 4 External Files Referenced (User Database, Product Database, Reviews Database, Message Database)

Record Element Types (RET)	**Data Elements**		
	1 to 19	20 - 50	51 or More
1 RET	Low (7)	Low(7)	Average (10)
2 to 5 RET	Low (7)	Average (10)	High (15)
6 or More RET	Average (10)	High (15)	High (15)

Summary of above analysis:<3

Element	**Complexity**	**Detail**
Inputs	Low	Gathering user account information
	High	Gathering product information
	Low	Preference and Constraints
Logical Files	High	Transaction Details
	Medium	User Account
Outputs	High	Display list of results matching user's criteria
	Low	Display merchant product listing
	Low	Display user account information
Inquiries	High	Selecting product according the user's criteria
	Medium	Selecting user transaction information
	Low	Selecting user account information
Interfaces	Medium	User Database, Product Database
	Low	Reviews Database, Message Database

Calculation of Unadjusted Function Points:

Characteristic	**Low**		**Medium**		**High**	
Inputs	2	× 3	0	× 4	1	× 6
Outputs	2	× 4	0	× 5	1	× 7
Inquiries	1	× 3	1	× 4	1	× 6
Logical Files	0	× 7	1	× 10	1	× 15
Interfaces	2	× 5	2	× 7	0	× 10
Unadjusted FP	27		28		34	
Total=L+M+H	89					

5.1.2 Adjusted Function Points

Influence Factors	Score	Detail
Data Communications	3	Application includes on-line data collection or TP (teleprocessing) front end to a batch process or query system.
Distributed Functions	4	Distributed processing and data transfer are online and in both directions.
Performance	3	Response time or throughput is critical during all business hours. No special design for CPU utilization was required. Processing deadline requirements with interfacing systems are constraining.
Heavily used	2	Some security or timing considerations are included.
Transaction rate	3	Daily peak transaction period is anticipated.
On-line data entry	5	More than 30% of transactions are interactive data entry
End-user efficiency	2	Four to five of the efficiency designs are included
On-line data update	3	Online update of major internal logical files is included.
Complex processing	1	Any one of the complex components
Reusability	4	The application was specifically packaged and/or documented to ease re-use, and the application is customized by the user at source code level.
Installation Ease	1	No special considerations were stated by the user *but* special setup is required for installation.
Operational Ease	1	Effective start-up, back-up, and recovery processes were provided, but no operator intervention is required (count as two items).
Multiple sites	0	User requirements do not require considering the needs of more than one user/installation site.
Facilitate change	3	Flexible query and report facility is provided that can handle complex requests, for example, *and/or* logic combinations on one or more internal logical files (count as three items).
Total score	35	
Influence Multiplier = Total score × 0.01 + 0.65 = 35 × 0.01 + 0.65 = 1.0		
Adjusted FP = Unadjusted FP × Influence Multiplier = 89 × 1.0 = 89		

Scoring (0 – 5)
0 = No influence
1 = Insignificant influence
2 = Moderate influence
3 = Average influence
4 = Significant influence
5 = Strong influence

5.1.3 Lines of Code

According to Quantitative Software Management (QSM), each Function Point requires on average 47 lines of code if the application is implemented using JavaScript.
Therefore, we have: Lines of Code = 89 FP × 47 LOC/FP = 4183 LOC

5.2 Efforts, Duration and Team Size Estimation

To estimate the effort and duration required for the project, we use function points as the basis to calculate Effort, Duration, Team size and finally the schedule. The estimates are expanded to account for project management and extra contingency time to obtain the total average effort estimates. From these averages, the duration of each work package in working days is estimated based on the following calculations.

- Working days include 5 days in a week.
- Effort = Size / Production Rate = (4183 LOC) / (45 LOC/PD)1 = 93 PD
- Duration = 3 × (Effort) 1/3 = 3 × (93) 1/3 = 13.6 Days
- Initial schedule = 13.6 Days / 5 days a week = 2.72 Weeks
- Team size = 93 PD / 13.6 D = 6.84 P = 7 Persons
- Working hours include 8 hours in a working day.
- Total person-hours (PH) = 93 PD × 8 hours = 744 PH

5.2.1 Distribution of Effort

1990's Industry Data	Work Package	Distribution	Estimates
Preliminary Design 18 %	Project Plan	9%	66.24
	Requirement Specification	9%	66.24
Detailed Design 25 %	User Interface	7%	51.52
	Technical Architecture	11%	80.96
	Data Modeling	7%	51.52
Code & Unit Testing 26 %	Code & Unit testing	21%	154.56
	Online Documentation	5%	36.8
Integration & Test 31 %	Integration & Quality Assurance	31%	228.16
	Extrapolated total effort		736
	2% for project management		14.72
	3% for contingency		22.08
	Total effort		772.8

These duration estimates are based on the assumption that each team member works an equal amount on any given work package.

[1] Lines of code per Person Day statistics based on Industrial Benchmarks, 1997: 31 LOC/PD for United States; 62 LOC/PD for Canada

5.3 Cost Estimates

Hardware:

Developer workstations:

7 - Lenovo Thinkpad X240	Total $3150.00
I7-4th Gen/ Windows 10 Pro	
8GB RAM	
256GB SSD	

Software:

Software License Provided by Third Party:

Microsoft Visual Studio Code	$0.00
Microsoft 365	$0.00
MongoDB Database	$0.00

Other Resources:

Staff:

7 Employees with 772.8 working hours with $16.00/hour	$12364.80

Stationary:

Paper, photocopying and other miscellaneous cost	$50.00

Total: $15,564.80

The customer will supply the required hardware and software necessary to run the MongoDB server and the E-Commerce web server. APT200 is not responsible in any way for supplying said systems. APT200's hardware and software responsibilities relate only to our own development needs to accomplish the project we have been asked to complete, and which has been described in the introduction section of this document. APT200 will also demonstrate the completed product.

6. Product Checklist

Project Deliverable	**Estimated Deadline**
Project Proposal	8 Sept 2022
Use Case Model	8 Sept 2022
Quality Plan	22 Sept 2022
Software Requirements Specification	22 Sept 2022
Risk Management	13 Oct 2022
Project Plan	13 Oct 2022
Prototype Demo	13 Oct 2022
Software Maintainability Report	27 Oct 2022
Configuration Management Plan	27 Oct 2022
Change Management Plan	27 Oct 2022
Release Plan	27 Oct 2022
Test Plan	10 Nov 2022
Test cases and Requirements Test Coverage Report	10 Nov 2022
Presentation Slides	10 Nov 2022

7. Best Practices Checklist

Practice	9
Document what we do; all documentation must be in a standardized format.	
Pay attention to requirements, check for ambiguity, completeness, accuracy, and consistency. The requirement documentation must contain a complete functional specification.	

Keep it simple. Complexity management is one of the major challenges. Strive to:

- Minimize interfaces between modules, procedures and data.
- Minimize interfaces between people, otherwise exponential communication cost
- Avoid fancy product functions, design as long as the functionality meets the customer requirements

Require Visibility. We must see what we build otherwise we can measure the progress and take management action. This includes: the manager must have good communication with his or her employees; require developers to make code available for review; review design for appropriateness.

Plan for continuous change.

- All manuals designs, test, source code should have revision numbers and dates revision history comments, change marks to indicate the changes
- New revisions should be approved before being made and checked for quality and compliance after being made
- Use a configuration management system and make processes
- Required maintenance

Don't underestimate. We must be careful to obtain accurate estimates for: time, effort, overhead, meeting time, and especially effort on integration, testing, documentation and maintenance.

Code reviews are a much more efficient method to find software defects. Plan and manage code reviews between team members

Software testing will use both black box and white box testing. It will involve unit, functional, integrating and acceptance testing.

8. Risk Management

Besides the general risk management, the following risks have been identified for the MerchantDice project:

More changes to requirements than anticipated
Impact Severity: High
Probability: 25%
Impacts: Depending on the stage at which changes occur, could range from needing to update the requirements documentation to needing to do a complete redesign.
Risk Reduction: Be rigorous in eliciting requirements. Make customers aware of potential repercussions of requirement changes.

Specification Delays
Impact Severity: High Probability: 15%
Impacts: Delay in finalizing the specification will push the schedule for all following stages of the project.
Risk Reduction: Monitor progress of specification carefully.

System size underestimated
Impact Severity: Moderate
Probability: 30%
Impacts: More work will need to be spent on design and coding; could negatively impact schedule.
Risk Reduction: Update estimates often as the project progresses.

System Performance Issues
Impact Severity: High
Probability: 5%
Impacts: Business activities would be negatively implicated due to performance issues encountered such as slow application performance.
Risk Reduction: Create a progressive benchmark to ensure that performance is consistent.

Staff leaving before project complete
Impact Severity: Extreme
Probability: 5%
Impacts: There would be more work for remaining employees, and any specialized skills or knowledge would be lost.
Risk Reduction: Offer benefits and incentives to staff.

Problems coordinating within group
Impact Severity: Moderate
Probability: 40%
Impacts: Members may be unaware of what is expected of them; managers may not be able to measure progress; portions of projects not completed.
Risk Reduction: Follow communication plans as documented in section 2.3

9. Quality Assurance

The project will achieve quality assurance by following the standard set by the company. The specific procedures and details shall be provided in the Quality Plan.

Specific test procedures and details shall be provided in the Module/System Test Plan.

In addition, MerchantDice shall make use of two testing methodologies:

- **Unit Testing** involves testing system components individually.
- **In-Place Testing** involves testing of the whole system as a unit.

Furthermore, these methodologies will be used to test these important aspects of MerchantDice:

- **System Function** will be tested to ensure that software flaws are eliminated, and
- **Algorithmic Function** will be tested to ensure that heuristic aspects of the project (such as user sorting preference) perform realistically to provide consistent and accurate results to the end users.
- **User Friendliness** will be tested to ensure that users of MerchantDice are able to navigate through the application routes intuitively without a high learning curve.

APT200's methodology makes broad use of realistic test cases. Detailed test data is an important part of the final project delivery. Although the MerchantDice application users are expected to furnish and enter data regarding products sold as well as other aspects involved in the MerchantDice system, APT200 shall provide a comprehensive and detailed subset of this data for testing purposes. APT200 will validate code and heuristic result ranking technology using realistic scenarios. In addition, extreme cases will be used to ensure that the system behaves correctly in degenerate cases.

10. Monitoring and Control

Many procedures are required in order to be able to successfully monitor the progress of a software project. Some of the most important are:

Quantitative measurement of resource consumption: Estimates of the Agent's resource requirements, primarily in terms of human resources, can provide a quantitative measurement of project progress when compared to progress in terms of project milestones. The percentage estimates of each milestone's resource requirements provided in this document allow for easy progress tracking.

Identification of major project risks: Early identification of major risks to the project allows for placement of preventative measures before problems can develop. Major risks have been identified in the Risk Management section of this document, along with the measures being taken to avoid them.

Regular reviews of project progress: Throughout the duration of the Airline Agent project, TLA shall meet weekly to review the progress of all project tasks, including management, planning,

analysis, development, and testing.

Timeline Planning and task decomposition: This document outlines an estimated timeline for the project. A reasonably accurate timeline can be assembled by hierarchically decomposing tasks into measurable subcomponents and estimating requirements for each. At the same time, this decomposition can assist in task assignment and balancing. Throughout the implementation phase, these subcomponents can allow for fine-grained measurement of progress. Project subcomponents and timeline estimates are included in the Estimates and Work Breakdown Structure sections of this document.