Project Proposal, and Plan

The Application Domain

We are proposing to develop a price comparison web application directed towards comparing prices of physical video games from several retailers. The system will include a web-based GUI, method of storing useful data, and several core features. The project's motivation is to find and compare prices of video games from various retailers, so consumers can save money and be informed on the changing prices of video games. Five features we aim to focus on include: tips on when the best time is to buy a game; the locations of the nearest retailers; notifications of when a product has reduced in price; the saving of user preferences when using the web app; and a comprehensive product search and filtering function.

One constraint to the development of the system we have considered is the localisation of the web app, concerning: currencies; postage, importation, and delivery; and region locked products. A second constraint is the number of sources we take data from, as with too few sources the web app would be ineffective in comparing prices, yet with too many sources the overall reliability of the sources may decrease. Furthermore, another constraint is the time limit we have to develop the web app.

We identified our target audience as consumers of video game products and aim to involve the target audience periodically through interviews with both focus groups and individuals representing our target audience, to fully assess that our work is aimed towards satisfying our requirements.

The Team

Our team's strengths largely reflect the knowledge that was developed during our first year of university. Jay's strengths include designing and writing classes using OOP principles, whereas Matthew's strength is with algorithms. Jamie's strengths are in backend development with databases while Sirena's strengths include problem solving in regards to all aspects of an application. Thomas's strengths include working with SQL and Python whilst Polys is confident with HTML and CSS. In terms of weaknesses, we have determined that the weakness of one individual can be compensated with the strengths of another. Jay struggles with UI design and development which is complemented by Matthew's strength, and Matthew's weakness is troubleshooting which is Sirena's strength. Furthermore, Thomas and Polys share a weakness working with Java, a language which Jay is confident with, and Jamie's weakness includes a lack of experience with JavaScript which both Sirena and Matthew can assist with in the development of this application.

Through the work completed last year we have collectively developed similar technical skills and understanding of both Java, Python, and HTML/CSS in regards to static websites. Most of the group have also developed knowledge of using MySQL for developing and managing databases. Despite these technical skills, most team members have not worked on larger projects that require multi-stage development processes. Therefore our knowledge of how to operate with longer term processes will be largely drawn from the INSE unit.

While most are familiar with some of the tools frequently used in software development, such as IDE's, code-level documentation like JavaDocs, each member also has specific experiences with other technologies in addition to Java, Python and HTML/CSS. Matthew has used Visual Basic, and JavaScript while Jay has experience with JavaDocs and MySQL. Sirena is familiar with JavaScript, SQL, some C++, and some Visual Basic whilst Jamie, Polys and Thomas have all had experience with SQL.

Responsibility within the group has been divided into 3 sub groups: GUI (comprised of Matthew and Thomas), Storage (Jamie and Polys), and Core Logic (Jay and Sirena). The sub groups have been split up in this way so that one person per group will be familiar with JavaScript which will be a key component for each component of our application to work in cooperation. Non-technical roles for this project have also been assigned to members wishing to take more responsibility; these roles include taking minutes of meetings (Jay), arranging team meetings (Sirena), updating attendance (Jamie), and submitting the files (Matthew).

Using the data collected from the team survey, we have determined that all team members are somewhat comfortable with taking a leading role, and subsequently have concluded that it would be appropriate for a member within each sub group to take a leading role in the development of each section to keep decision making both efficient, and relevant to the tasks involved. While the team survey results [2] captures our initial impressions on the roles we may take, through the team meetings we were able to develop a stronger idea on how best to assign team roles.

The Process

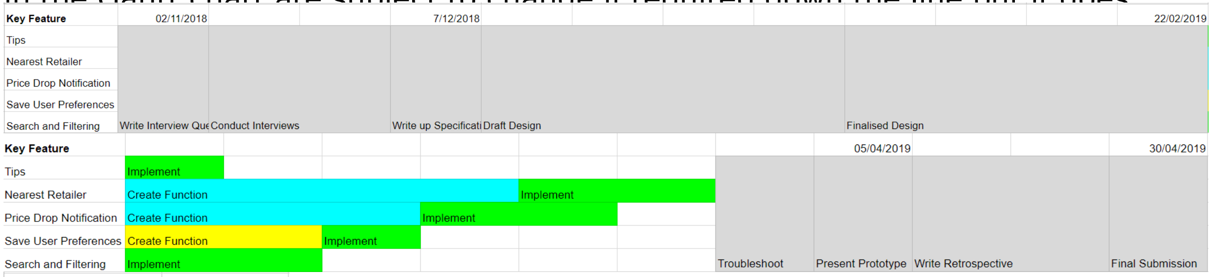
Before setting out and completing tasks, we will clarify which member is responsible for ensuring a section of the work is completed. This means the work load is distributed evenly throughout the group, and no one person ends up doing all the work while others contribute nothing. The weekly meetings allow this process to be kept in check by members being able to update each other on their individual progress. We decided upon following a waterfall model design process due to its linear design approach that allows clarity to each member on which stage of the development process is underway.

To determine these roles, a questionnaire [1] was designed and answered by each team member that focused on learning what prior experience each person had with software development and to determine what area they felt most comfortable working in.

The main project milestones revolve around the submission dates stated in the coursework specification. We aim to complete the work for these dates a week in advance to ensure no last-minute rushes/late submissions. Other milestones include the end goal of being able to produce a website that outputs correct data after a search request has been made. During meetings we will discuss the progress that has been made in our various sections of the project, present ideas for improvements, etc. At the start of most meetings we will set an agenda in order to keep activity focused. We will record attendance at meetings to keep track of everyone's contributions and to ensure everyone is caught up with information shared in those meetings.

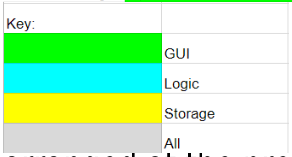
We concluded that a Gantt chart [3] would be valuable when considering the timeline of different tasks throughout the process. It enables the ability to prioritise the completion of specific tasks, and being able to plan ahead to each task is completed on schedule. The chart lists the tasks to be completed on the vertical axis, and time intervals on the horizontal axis. The width of the each of the horizontal bars in the graph shows the duration of each activity. We colour coded each subteams tasks to make it clear what tasks each team was tackling e.g. the GUI team would focus on specific tasks within the implementation stage.

Due to the long development timeline of a project such as ours, the dates set out in the Gantt chart are subject to change if required down the line but it does



Key Feature	02/11/2018			7/12/2018								22/02/2019
Tips												
Nearest Retailer												
Price Drop Notification												
Save User Preferences												
Search and Filtering	Write Interview Que	Conduct Interviews		Write up Specificati	Draft Design				Finalised Design			

Key Feature								05/04/2019			30/04/2019
Tips	Implement										
Nearest Retailer	Create Function				Implement						
Price Drop Notification	Create Function			Implement							
Save User Preferences	Create Function		Implement								
Search and Filtering	Implement						Troubleshoot	Present Prototype	Write Retrospective		Final Submission



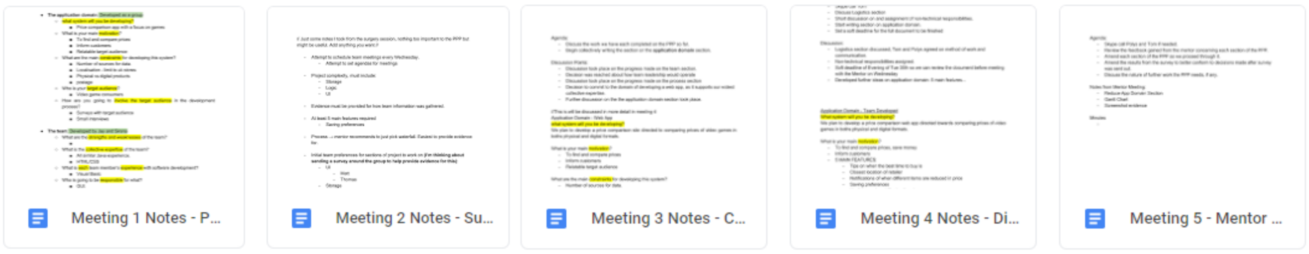
Key:	
🟩	GUI
🟦	Logic
🟨	Storage
▨	All

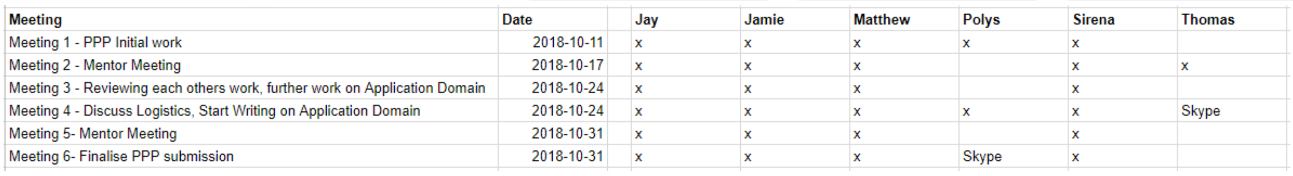
usually scheduled on Wednesdays and further meetings are arranged at the previous meeting; team members are notified the exact times by the person responsible for arranging and booking the meeting space (Sirena) via email/WhatsApp.

Software packages chosen for facilitating communications across the team include WhatsApp, email, Google Drive and Skype. WhatsApp is the primary communication tool used for arranging meetings and discussing project queries with email serving as an auxiliary method in the event the former is unsuitable for any reason. Skype is used to facilitate meetings where a team member cannot make the meeting in person and can also be used when a meeting area cannot be found. Slack was also considered but the aforementioned software suites fulfils the needs of the team without requiring additional time to learn a new software package. However, further along in the implementation of our application, Github will be used once the coding process has begun. Google Drive is used as a group accessible file repository allowing files to be shared and edited by the team. It also serves as a offsite backup ensuring redundancy of the project and as a method for recording meeting outcomes in the form of minutes.

In the event of a work-related conflict the issue should be discussed by the entire team for possible resolutions if after all possibilities have been explored and the group remains divided on how to proceed a vote should be taken. In the event of a stalemate, a random numeric generator (RNG) will be used to pick an option this will end the conflict allow the team to proceed and prevent blame being assigned if the option chosen by RNG doesn’t solve the cause of the original problem.

An Excel spreadsheet will be used to keep track of the attendance at the beginning of each meeting. Also an online copy will be stored on Team Drive for everyone at the end of each session. There will be records also for online attendance via Skype for people who can’t attend the meeting. Shown below are screenshots for meeting notes and agendas stored on the Team Drive as well as





Meeting	Date	Jay	Jamie	Matthew	Polys	Sirena	Thomas
Meeting 1 - PPP Initial work	2018-10-11	x	x	x	x	x	
Meeting 2 - Mentor Meeting	2018-10-17	x	x	x		x	x
Meeting 3 - Reviewing each others work, further work on Application Domain	2018-10-24	x	x	x		x	
Meeting 4 - Discuss Logistics, Start Writing on Application Domain	2018-10-24	x	x	x	x	x	Skype
Meeting 5- Mentor Meeting	2018-10-31	x	x	x		x	
Meeting 6- Finalise PPP submission	2018-10-31	x	x	x	Skype	x	

the attendance of team members up until the due date of this deliverable.

Appendix

1. Team survey - https://goo.gl/A4oWSE
2. Team survey results - https://goo.gl/TGnnLd
3. Gantt chart - https://goo.gl/3jMsDr

System Requirements Specification

0. Preface

Revision	Date	Description
1	2018-11-28	Revision includes requirements and requirement specifications gained from conducting interviews.
2	2018-12-05	Revision includes additional requirements and requirement specifications learned from conducting a focus group and observational study.
3	2018-12-06	Revision includes the validation of requirements gathered.
4	2018-12-07	Revision works upon feedback gained from Mentor feedback.

1. Introduction

Our web application aims to provide a unique way for the target audience (customers interested in purchasing games) to compare prices of video games from a large group of retailers within a single site. Other sites, such as gamecrawler.co.uk, provide a similar service to what we intend to offer but miss many of the features the feedback from focus groups and interviews has shown to be of importance to users.

The ability to view historical pricing data of a game is a feature that allows the customer to make an informed choice with statistical evidence to show they are getting a good deal. A function that was identified as useful to potential customers is the ability to filter search results via user entered constraints. Many popular comparable sites already do this, such as Amazon.co.uk and game.co.uk, so the user should be able to tailor results in a way they are familiar with. A login page is another system function that would allow the user to personalise their experience in a way similar to Amazon, being able to favourite game genres and suggest games of interest using this.

1.1 Glossary

Downtime	Time during which a system is out of action or unavailable for use.
Functional Requirement	Statements of services the system should provide including: how the system should react to inputs, how the system should behave and what the system should not do.
GUI	Graphical User Interface.
Methodology	Methodology is the systematic, theoretical analysis of the methods applied to a field of study.
Non-functional Requirement	Constraints on the services or functions offered by the system.
Prerequisite	Something that is required as a prior condition for something else to happen or exist.

SRS	System Requirements Specification.
Uptime	Time during which a system is in operation.
User	Intended person utilising the final product.
Validation	Data validation is a process that ensures the delivery of clean and clear data to the programs, applications and services using it.

2. Requirements Elicitation Methodology

2.1 Interview Methodology

One method we used to gather information for our system requirements was conducting interviews. These interviews followed a semi-structured format, meaning that there was a pre-written set of questions for interviewers to ask, but they were also free to ask additional questions when there was reason to, such as to clarify points made by the interviewee or to gain more insight into a point that was made. The questions that we wrote beforehand were all open questions, as this would allow us to get a large variety of answers and would reduce the impact of any biases we had on the response we got.

We recorded the responses to the prepared questions in a Google form [I], with a section at the end for miscellaneous information that was not given as an answer to a given question. We have also summarised the responses [II] as various graphs [V] in order to easily ascertain what requirements user's highly requested.

2.2 Focus Groups Methodology

The focus group [III] took place on November 23rd. We had multiple people present and our main goal was to stimulate discussions reflecting on the answers we gained from the interviews. Two of the main features that were asked about are the specifics of the GUI the focus group want to see and how the notifications should work so that we gain a consensus. During the focus group there were no leading questions asked, but instead we attempted to gain insight on the opinions of potential users by generating discussion on topics concerning the system we intend to develop. When we had all we needed we determined a consensus.

2.3 Observational Study Methodology

When conducting the observational study, we set the participants a distinct goal that mirrored an expected use case for our web app and encouraged use of the type of functionality that our web app seeks to achieve. By noting the way in which a participant would carry out their task (by looking at the number of websites used, the types of search options used, and the way in which the user navigates a web site) we were able to determine the flaws and strengths of sites with similar objectives to ours. While observing the participants, we made sure to take notes [IV], as they utilised various websites, that could effectively and accurately illustrate the process the participant underwent to achieve their set task.

2.4 Requirements Validation [VI]

By contacting the participants from our interviews, focus group and observational study, we were able to validate our requirements. Through discussing the user requirements definitions we generated, with these potential users, we could affirm that both the Team and the participants are in agreement on requirements of the system.

3. User Requirements Definitions

3.1 ***Site should have a monthly uptime of 99.99%, which allows for 4.38 minutes of downtime per month.***
Reference: Responses to interview Question 1.

3.2 ***Site should be accessible from multiple types of common device, including: laptops, PCs and mobile phones.***
Reference: Responses to interview Question 2

3.3 ***User should be able to decide the conditions under which they are notified, including: time frequency, during seasonal sales events, after new product releases.***
Reference: Responses to interview Questions 3.

3.4 ***User should be able to decide the medium through which they are notified, e.g. via: email, SMS message, or push notification.***
Reference: Responses to interview Questions 4.

3.5 ***Site should display seasonal sales events on the homepage.***
Reference: Interview question 1, Focus Group section 2

3.6 ***Site should save to users search history if opted in.***
Reference: Interview question 4

3.7 ***Site should save the types of consoles the users selects***
Reference: Interview question 4

3.8 ***Site should save the users favourite genre.***
Reference: Interview question 4

3.9 ***Site should suggest games based on the user's search history, if set by user.***
Reference:Interview question 4

3.10 ***Site should filter search results by: Price, Platform, Most popular, genre, release year, product title alphabetical order***
Reference:Interview question 5

3.11 ***Site must be easily accessible, and usable by all potential users.***
Reference:Responses to interview Question 6.

3.12 ***Site must have a login option available, accessed away from the main body of the home page.***
Reference: Responses to interview Question 4.

3.13 ***Site should follow conventions established by current services.***
Reference: Responses to focus group Question 5.

3.14 ***Individual game page should include reviews, both reviewer opinions and star rating.***
Reference: Responses to interview Question 6.

3.15 ***Site homepage should be minimalist but not overbearing***
Reference: Responses to focus group Question 4.

3.16 ***Site should use recognisable logos that a user can easily identify, aiding navigation.***
Reference: Responses to interview Question 6.

3.17 ***Product page should display information on the stock of the product.***
Reference: Responses to focus group Question 3.

3.18 ***User wants distinct experiences when using PCs and phones***
Reference: Focus Group section 1.

3.19 ***Each product should have a distinct page***
Reference: Responses to Interview question 6

3.20 ***Site should make effective and efficient use of space***
Reference: Focus Group section 4

3.21 ***Popular products should be presented in an eye catching way using animation/slideshow on main page.***
Reference: Focus group section 4

3.22 ***Site should avoid unpopular design choices, such as making use of pop-up dialogues.***
Reference: Focus group sections 4 and 6

3.23 ***Site should indicate to user the nearest retail locations with stock.***
Reference: Focus Group section 6, Observational Study section 2 and 3

3.24 ***Site should provide usability support, FAQ.***
Reference: Focus Group section 6

3.25 ***Site should present historical pricing data***
Reference: Observational Study Sections 4 and 6

4. System Requirements Specification

4.1 Functional Requirements

Function	The conditions under which the user is notified, and the medium through which the user is notified can be set by the user.
Description/Rationale	This enables the user to set the conditions for when they are notified, in order to best satisfy each user's specific preferences. This gives the user the option to determine the means by which they are notified.
Input	User selection of an option determining notification frequency characteristics. User selection of a notification medium.
Output	A notification delivered to the user, concerning information the user desires.

Action performed	Once the user determined condition has been met a notification should be sent to the user through the medium that the user has selected.
Prerequisites	The user has determined their notification conditions, and prefered notification medium.
Error handling	Error can be handled by utilising data validation, ensuring that the user has provided correct inputs such that a correct notification can be delivered.
Validity test	Measuring that a notification is received by a test user under given conditions and satisfies expectations
Source/reference	3.3, 3.4

Function	To inform users of seasonal sales.
Description/Rationale	If users are aiming to save money using this app, it may be useful to inform them when sales are on.
Input	Information from shopping sites about seasonal sales.
Output	Text naming the outlet holding the sale and a description of the sale
Action performed	None.
Prerequisites	The stores having the sales that will be announced need to have a website. This website needs to have the same prices as the physical store.
Error handling	If there is an error, the text will not appear.
Validity test	We will run the web app and see if the messages about seasonal sales appear. We will then compare the message to the one on the outlet’s website to ensure that the information is accurate.
Source/reference	3.5

Function	Save users’ search history
Description/Rationale	When a user begins typing in their query, previous search terms will appear in a drop down menu below the text box. Suggestions will disappear if the user enters characters that do not appear in those search terms. Users may want to look at the same product multiple if they plan on buying a product in the future and/or want to see if the price changes.
Input	The users search query
Output	N/A
Action performed	Storage
Prerequisites	There needs to be an array that contains a user’s previous search terms
Error handling	If there is an error, no suggestions will appear.
Validity test	We will search for several terms, then see if they appear as suggestions when we click on the search bar afterwards. We will then enter a previous search term to ensure that that suggestion stays while the other ones are removed. We will then enter a new query to ensure that all suggestions are removed.
Source/reference	3.6

Function	Save the types of consoles users select
Description/Rationale	Users are unlikely to buy or get rid of consoles often, so it is very likely that the app will be more convenient to use if they do not have to select which consoles they want games for every session.
Input	Console radio boxes
Output	Selected radio boxes
Action performed	Storage
Prerequisites	This is likely to require the user to either allow the app to store data locally or use cookies.
Error handling	If there is an error, all consoles will be deselected.
Validity test	We will load up the app twice with no consoles selected. Both times, the radio boxes should remain empty. We will then select a console, close the app and open it again. The selected radio box should remain selected when the app is reopened. We will do this for every individual console and for all consoles at once.
Source/reference	3.7

Function	Site should filter search results by: Price, Platform, Most popular, genre, release year, alphabetical
Description/Rationale	Allow the user to customize search results using filters to save time when browsing through lists of games.
Input	User selects filter from Price, Platform, Most popular, genre, release year, alphabetical
Output	Filtered search results returned
Action performed	N/A
Prerequisites	User must choose to filter search results otherwise all games are returned
Error handling	N/A
Validity test	Test whether a search returns the correct games when a filter is applied
Source/reference	3.10

Function	Site should suggest games based on the user’s search history, if set by user.
Description/Rationale	This suggests games to the user that are relevant to their past searches.
Input	User search history
Output	Relevant games shown to user
Action performed	N/A

Prerequisites	User must enable save search history.
Error handling	N/A
Validity test	Check user history is saved correctly, and games suggested are relevant
Source/reference	3.9

Function	Site should save the users favourite genre.
Description/Rationale	This enables users to quickly navigate through games of interest by allowing searches by the users favourite genre
Input	User selected genre of choice
Output	Enable the ability to filter games to only show that genre.
Action performed	N/A
Prerequisites	The user has determined their genre of choice.
Error handling	N/A
Validity test	Test if searching by favourite genre returns accurate results
Source/reference	3.8

Function	Site must have a login option available, accessed away from the main page.
Description/Rationale	This enables the greatest user accessibility by saving their unique preferences.
Input	username/password
Output	account
Action performed	Checking if the account details match with the system.
Prerequisites	Must be unique for each user with the preferences they choose.
Error handling	Yes, if the details entered are wrong an error message should be appear.
Validity test	This requirement can be tested by using random Id/passwords to check if the system is secure..
Source/reference	3.12

Function	Individual game page should include reviews, both reviewer opinions and star rating.
Description/Rationale	This allows users to make an informed decision on whether to purchase a new game based on the recommendations and opinions of reviewers. By providing both opinions and star ratings, this allows for both an in-depth review of a game but also provides a concise rating at a glance.
Input	User selects a product and is redirected to the product’s specific page.
Output	User is presented with both written reviews and star ratings.

Action performed	N/A
Prerequisites	Game must have reviews available elsewhere, therefore very recent games may not have reviews immediately following release.
Error handling	Error due to unavailable review data.
Validity test	Check various product pages to see if game reviews are being displayed.
Source/reference	3.14

Function	Product page should present the nearest retail location at which a product is in stock.
Description/Rationale	Users have suggested that when searching for games they would like to see the current stock options at nearby locations that sell the required product so that their purchase of a game as convenient as possible.
Input	A product, a location, an acceptable distance radius
Output	The retail locations within the given radius that have the product in stock.
Action performed	Use GUI to produce output results and logic to determine viable retailers.
Prerequisites	Use inputs. Map integration.
Error handling	Error message displayed location input by use is invalid
Validity test	If the site can output correct results from the necessary inputs the function can be considered valid
Source/reference	3.17 & 3.23

Function	Site should provide historical product pricing data
Description/Rationale	Users have requested that, in order to accurately gauge how cheap a game could feasibly be priced, historical pricing data should be available so that an informed decision can be made on whether a game is currently at a reasonable price.
Input	Product title, records of the product’s historical price data.
Output	Graphical representation of a product pricing history, presented on the product’s page.
Action performed	Historical pricing data, taken from a database, should be displayed to user in a line graph format, live. Price points can be taken periodically and the graphs should change to represent the current prices.
Prerequisites	Methods to take and store price information from other sites, to a database.
Error handling	If there is a change in the way the data gathered becomes accessible, such that errors occur, manual configuration of methods may need to take place.
Validity test	If a product page can display, graphically, the historical pricing data of a product in a way that is easy to comprehend, the specification can be deemed valid.
Source/reference	3.25

4.2 Non-functional Requirements

Function	The site must maintain an uptime of at least 99.99%
Description/Rationale	This requirement ensures that any user will be able to access the site at any desired time.
Prerequisites	The site must be accessible by users.
Validity test	Site uptime can be tested by recording the amount of time for which the site is accessible and not accessible, and then comparing the two values over time.
Source/reference	3.1

Function	Site should be accessible from multiple types of common device.
Description/Rationale	This enables the greatest user accessibility as it does not restrict the users access due to using a particular device.
Prerequisites	Site must be implemented in such a way that can be presented across several different types of device.
Validity test	This requirement can be tested by evaluating the quality of the presentation of the site when being accessed by a series of common devices.
Source/reference	3.2

Function	Site must be easily accessible, and usable by current services.
Description/Rationale	Must be easy to use.
Prerequisites	Site content must be presented .
Validity test	This requirement can be tested by evaluating the quality of the presentation of the site when being accessed by a series of common devices by different people.
Source/reference	3.11

Function	Site should follow conventions established by current services.
Description/Rationale	Filtering sorting on the top or left, an obvious search bar, with a drop down menu. Inclusion of familiar icons: magnifying glass for searches, silhouette sign in button, heart for favoriting.
Prerequisites	Site must be functional for all users.
Validity test	We will run multiple times all the features to make sure that everything works smooth.
Source/reference	3.13

Function	Site homepage should be minimalistic in design and not overbearing.
Description/Rationale	Users have requested the homepage to not be cluttered while displaying the relevant information needed to properly navigate the site.

Prerequisites	Site must be clear and concise whilst only displaying all the relevant and required information
Validity test	Requirement can be tested by once again gathering user's opinions on what could be improved until a final design has been approved.
Source/reference	3.15

Function	Site should use recognisable logos that a user can easily identify, aiding navigation.
Description/Rationale	Users rely on visual stimuli such as recognisable logos to know where to navigate to on an unfamiliar application.
Prerequisites	Logos should be chosen based on how common and widely recognisable they are.
Validity test	Observe users using the site to see whether the logos are instantly recognisable and easy to use.
Source/reference	3.16

Function	The site should follow effective graphic design practices and principles.
Description/Rationale	Such that the site uses familiar ideas that the user can understand
Prerequisites	Graphical design can be established after initial abstract system design modelling has taken place
Validity test	If feedback from potential users on the design of the site is positive, the specification can be deemed valid
Source/reference	3.20, 3.21, 3.22

Function	Site should follow established web standards and practices.
Description/Rationale	By following modern conventions such as using HTML5, CSS3, and Javascript we can ensure that that site can be deployed in the most accessible, efficient, and easy-to-maintain way.
Prerequisites	Design process should be complete as this specification applies most to the implementation process.
Validity test	If the site's pages can pass web standards validation testing, the specification can be deemed valid.
Source/reference	3.18, 3.19, 3.24

Appendix

I. Interview prompt questions - https://goo.gl/forms/WDbIZEcidIBUG4Tn1
II. Interview responses- https://goo.gl/A69o3u
III. Focus group notes - https://goo.gl/vcsD1u
IV. Observational study notes - https://goo.gl/MdGNjp
V. Interview response diagrams- https://goo.gl/d5KU4Y
VI. Requirement validation interview responses - https://goo.gl/k6eC6k

Design Documentation

Use-case Modelling

Scenario Specification

Find locations of nearest retailers with stock.	
Actors	User, MapManager, StockManager
Description	From the product page the user is using, the system can return retail locations on a map where stock is available. The products' webpages have access to scripts which can access data from the product and retailer databases to display the stock quantity of a product at a specific retailer.
Data	Product (implicitly gained from the page the user is using), User postcode (gained from user entry), users desired radius limit (gained from user entry).
Stimulus	User entering postcode and radius limit into the map GUI feature on a product page.
Response	System should display retailers on a map within a user specified range that have the product specified in stock.
Comments	By hosting the retailer map GUI feature within the product web-page, we can reduce the need for input validation by assuming that the user wants to check the stock at retailers for the product of the page they are accessing the map.
Error cases	Error cases will occur when: the user enters an invalid postcode; the stock for product is not found; the stock for product is found, but out of user specified range.

User modifies and saves user preferences	
Actors	User, User database
Description	How the system processes the changes in the user preferences database from the users choice of preferences selected on the account settings page.
Data	User preferences selection.
Stimulus	User selects and saves option.
Response	System updates user database with user new user preferences.
Comments	Friendly system of choices. Easy to understand.
Error case	User preferences failed to save, user will have to re-enter preferences.

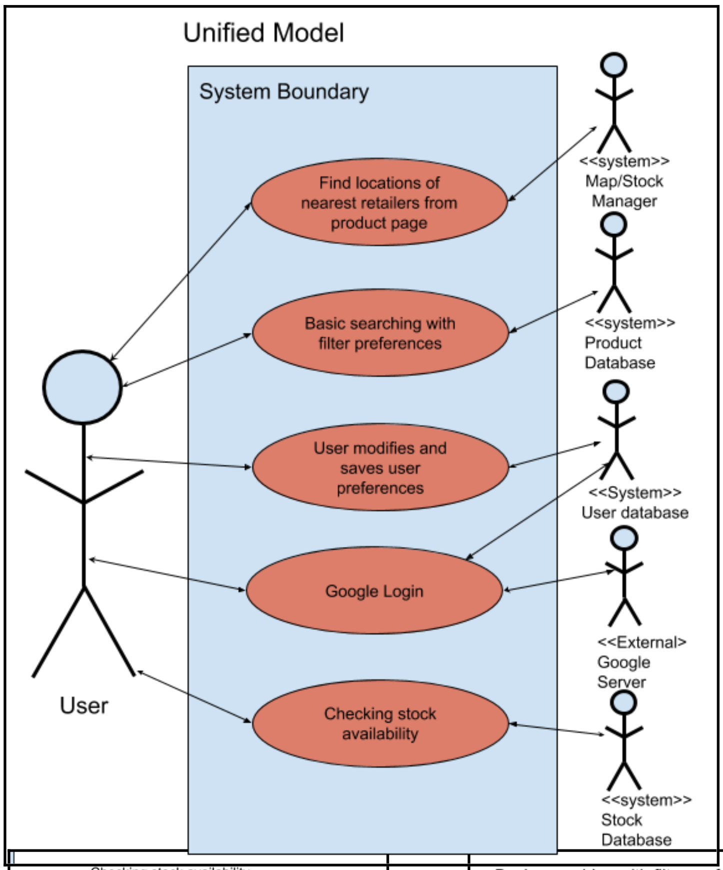
Log in through Google	
Actors	User
Description	A user can log into this web app using a Google account. A logged-in user is able

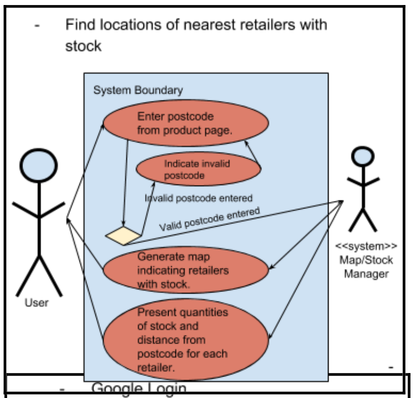
	to save changes to their preferences and previous changes are loaded.
Data	The user’s email address and password
Stimulus	User command issued by shopper
Response	Reload the page with a version that gives the user access to saved details and preferences from previous sessions.
Comments	Even without logging in, a users preferences can be saved during a browser session using cache.
Error case	If the user inputs an invalid email address and/or password, then they will be returned to the login page with an error message.

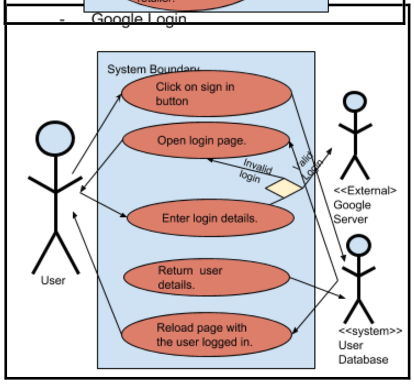
Check stock availability via search	
Actors	User, GameDatabase
Description	A user can search via the sites search bar for any game. If the game is within the database, the search returns the relevant game along with its current stock status (In or Out). If the game is not within the database, an error page telling user the game is not available is returned.
Data	A user search for a game
Stimulus	User clicking the search button loading results of search
Response	Output the relevant game after page is reloaded if it exists in database. Otherwise give error saying game can not be found.
Comments	System only gives overall stock status across all retailers, not individual retailers.
Error case	Game is not in database so cannot be returned to user. An error page is displayed saying the search gave no results is returned to the user.

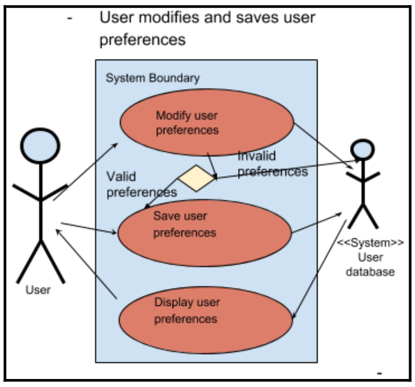
Basic searching with filter preferences	
Actors	User, Database Enquiry Manager
Description	A user may enter search parameters and filter preferences into a search bar on any page and the system will check whether there are matching products and display them for the user.
Data	Search parameters and filter preferences.
Stimulus	User enters search parameters and filter preferences into the search bar and filter checkboxes respectively.
Response	These parameters are then converted into SQL queries to check against the database. The system then returns any matching products.
Comments	System returns matching products including if there are no matching products, doesn’t account for misspelling.
Error case	Invalid search parameters have been entered or if no matching products have been found in the database.

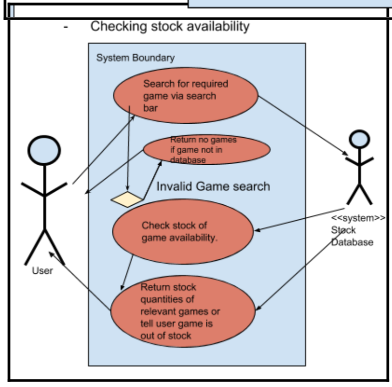
Use Case Diagram

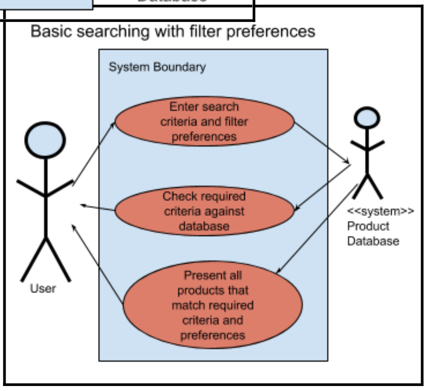

Unified Model
System Boundary
Find locations of nearest retailers from product page
Basic searching with filter preferences
User modifies and saves user preferences
Google Login
Checking stock availability
User
<<system>> Map/Stock Manager
<<system>> Product Database
<<System>> User database
<<External> Google Server
<<system>> Stock Database


- Find locations of nearest retailers with stock
System Boundary
Enter postcode from product page.
Indicate invalid postcode
Invalid postcode entered
Valid postcode entered
Generate map indicating retailers with stock.
Present quantities of stock and distance from postcode for each retailer.
User
<<system>> Map/Stock Manager






- Checking stock availability
System Boundary
Search for required game via search bar
Return no games if game not in database
Invalid Game search
Check stock of game availability.
Return stock quantities of relevant games or tell user game is out of stock
User
<<system>> Stock Database


Basic searching with filter preferences
System Boundary
Enter search criteria and filter preferences
Check required criteria against database
Present all products that match required criteria and preferences
User
<<system>> Product Database

For the sake of readability and interpretability, the “Unified Model” demonstrates the use cases of the system at-large, but also diagrams with more detail on each use case are included, each of which can be better seen in the folder linked in the appendix.

Sequence Diagrams

Components and Actors Identification

Find locations of nearest retailers with stock	
Actors: User.	**Components:** Map in product page, Map Manager, Stock Manager, Product Database, Retailer Database.

Basic searches with filter preferences	
Actors: User.	**Components:** Search bar on page, Database Enquiry Manager, Product Database.

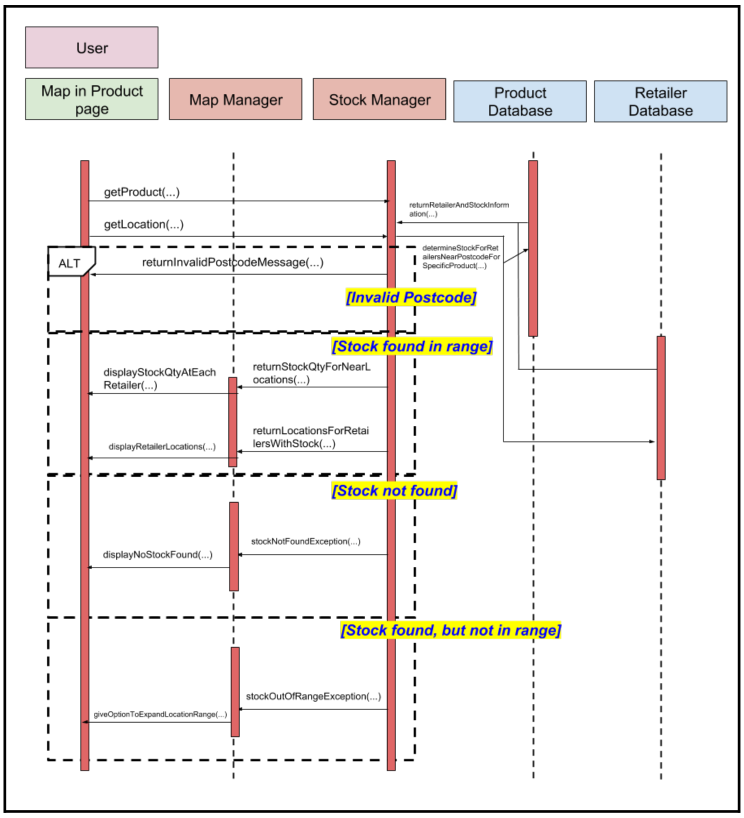
User modifies and saves user preferences	
Actors: User.	**Components:** Account Settings Page, User Account Manager, User Preferences DB Server

Google Login	
Actors: User.	**Components:** Login button on page, Google API Login Manager, User Account Manager, User Preferences DB Server.

Checking stock availability	
Actors: User.	**Components:** Search bar on page, Stock Manager, Database Enquiry Manager, Product Database.

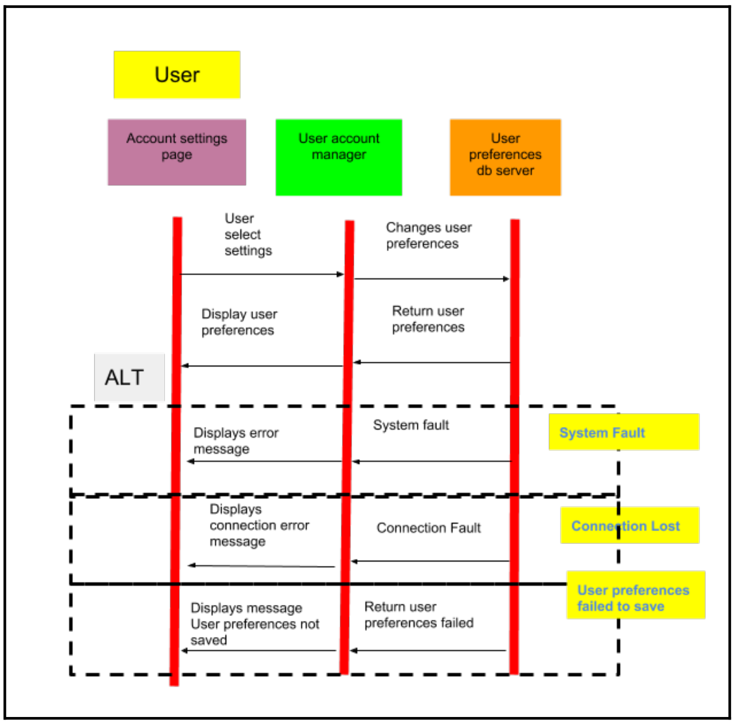
Diagrams

Find locations of nearest retailers with stock

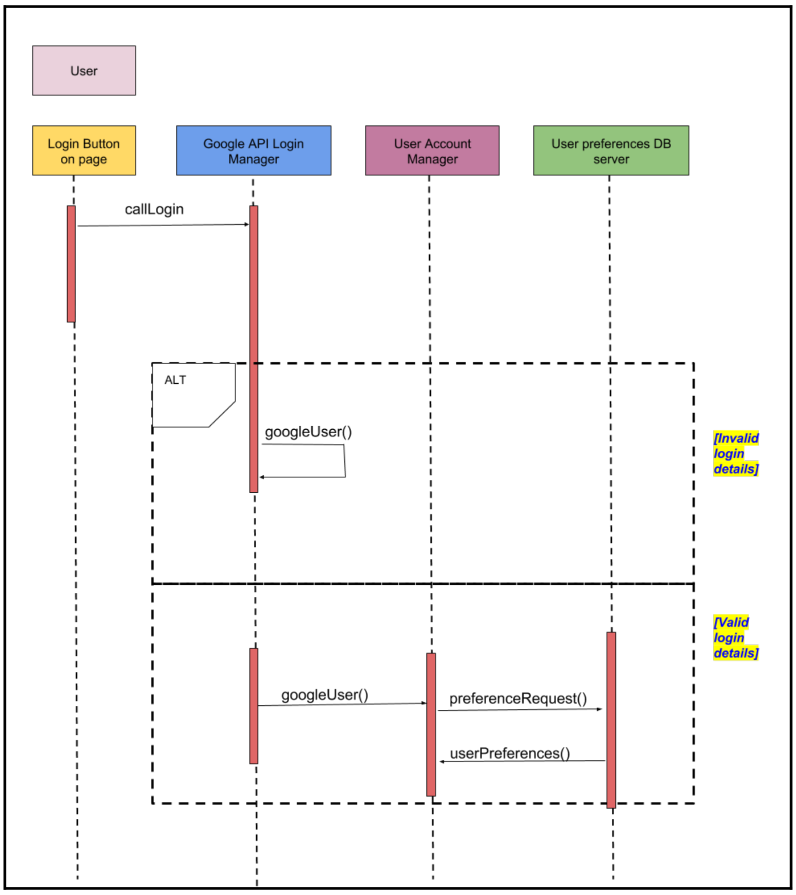


User
Map in Product page
Map Manager
Stock Manager
Product Database
Retailer Database
getProduct(...)
returnRetailerAndStockInformation(...)
getLocation(...)
determineStockForRetailersNearPostcodeForSpecificProduct(...)
ALT
returnInvalidPostcodeMessage(...)
[Invalid Postcode]
[Stock found in range]
displayStockQtyAtEachRetailer(...)
returnStockQtyForNearLocations(...)
returnLocationsForRetailersWithStock(...)
displayRetailerLocations(...)
[Stock not found]
stockNotFoundException(...)
displayNoStockFound(...)
[Stock found, but not in range]
stockOutOfRangeException(...)
giveOptionToExpandLocationRange(...)

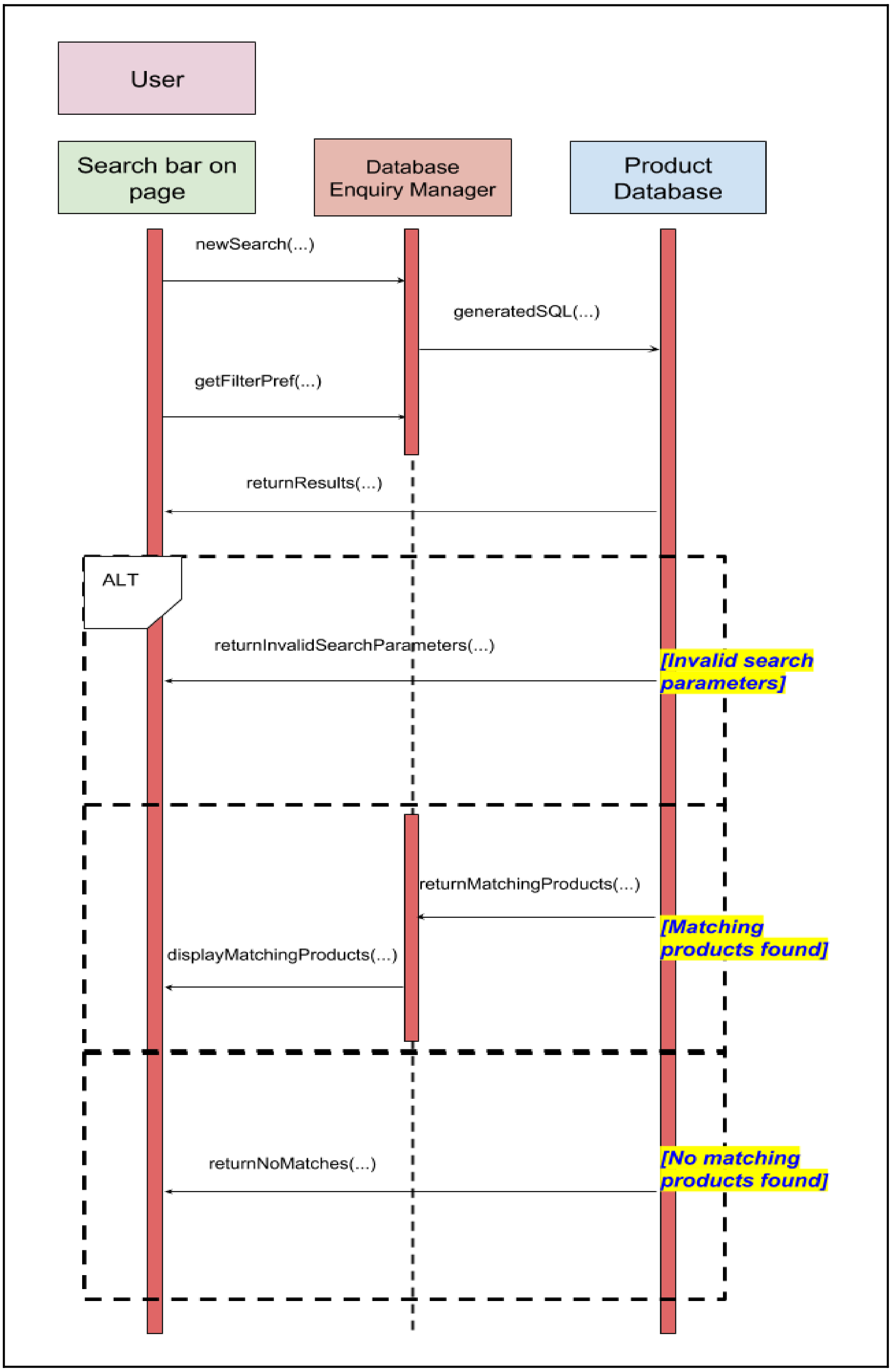
User modifies and saves user preferences



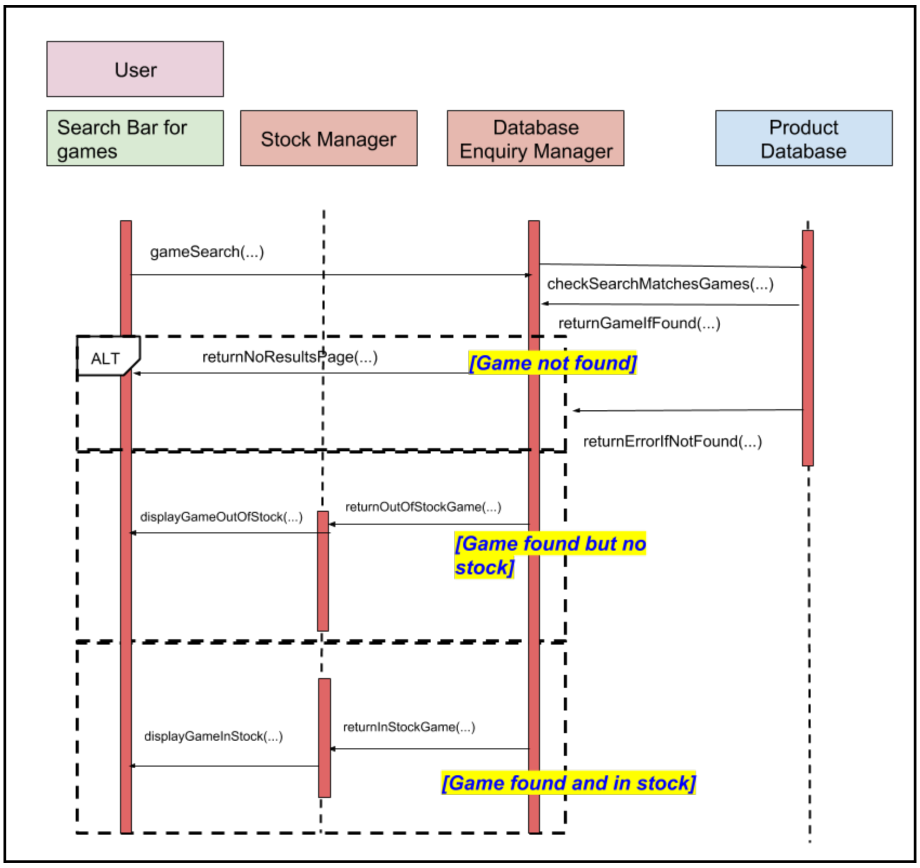
Google Login



Basic searching with filter preferences


User
Search bar on page
Database Enquiry Manager
Product Database
newSearch(...)
generatedSQL(...)
getFilterPref(...)
returnResults(...)
ALT
returnInvalidSearchParameters(...)
[Invalid search parameters]
returnMatchingProducts(...)
[Matching products found]
displayMatchingProducts(...)
returnNoMatches(...)
[No matching products found]

Checking stock availability



Architectural Modelling

Architectural Patterns Trade-offs

Object/Broker Trade Offs

Object/Broker (O/B) architectures share a large number of commonalities with the Client/Server (C/S) architecture, however one advantage O/B has over C/S is that the broker component adds a layer of distance between clients and the servers which means that servers are less vulnerable to attacks, as the broker can monitor and evaluate whether client connections are malicious. Additionally, O/B allows for clients not needing to know which specific server is being used which gives the system the freedom to better manage client access of servers by distributing services across multiple servers. This can be effective at managing high traffic periods however the needs of our system would not necessitate the use of a broker between the client and the server.

Layered Architecture

Using a layered architecture would allow for replacing layers with re-implemented if they are compatible, which enables an easy approach to maintaining and upgrading a system over time, while also helping to redundant services which can ensure dependability across extended project life cycles. However layered architectures can also bring the difficulties of distinguishing between the discrete layers of the model when developing system components and issues surrounding efficiency due to interpreting services across multiple layers. A layered approach could be effective in describing the structure of a subsystem within our system such as a server, however, as a complete system architecture a layered approach provides limitations that could restrict the development of our system aimed at satisfying client requirements, which could require communication between two different layers separated by another layer (i.e. a web-server hosting HTML files on the GUI level without going through the logic layer).

Event-based Architecture

An event-based architecture relies on components reacting to externally generated events and only communicate with other components through events. This means that despite evolution being made easy since adding new agents is simplistic, performance is a concern since the system doesn't produce an output unless an event triggers the system. Therefore, a build up of users accessing the system all at once may use up more resources than available resulting in a deadlock. Our system is better suited to a client/server architecture due to the system being broken into more manageable components and general functionality is available to all clients.

Repository Architecture

By using a system modelled after the Repository Architecture (RA), it can be ensured that data is propagated and kept consistent across components as data is stored in one place. This brings the cost of the repository being a single point of failure and the difficulty of maintaining a consistent repository across multiple machines. As we anticipate that our system will consist of many interconnected components that will need to share data, it would be inefficient to do this through one centralised repository, but instead let these components directly communicate.

Pipe and Filter Trade Offs

The pipe-and-filter (PF) architectural pattern is similar to the client/server (C/S) architecture due to components in both patterns not needing to know about other components. One benefit of PF over C/S is that it's easy to understand since the workflow process follows a typical business structure. However, a drawback of PF would be that the format for data transfer needs top be agreed between communicating components and each component must parse its input and unparse its output to the agreed form; this is an additional layer of data manipulation which reduces the efficiency of PF.

Identifying and Justifying Architectural Patterns

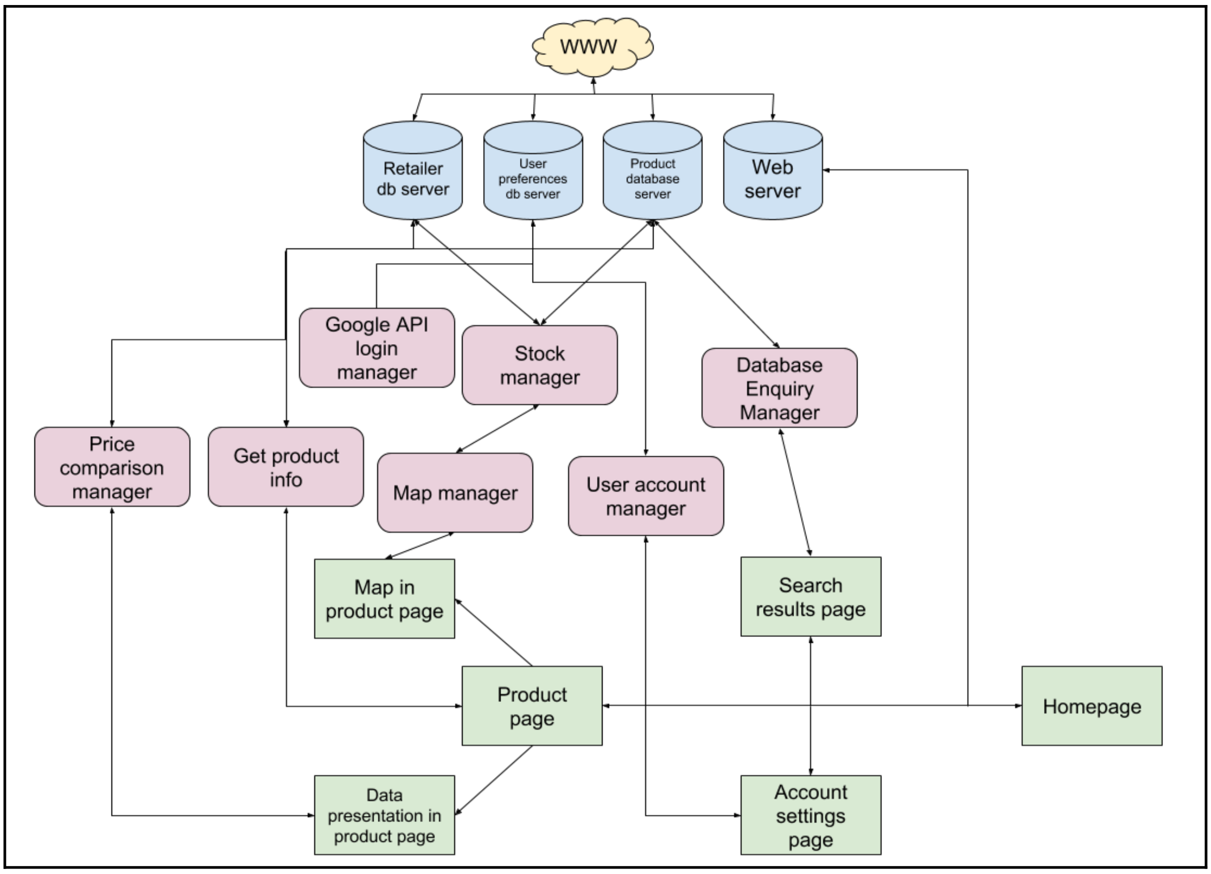
We decided upon using a client server object oriented architecture. In this architecture the servers provide services while the clients request and use these services. This suits the design of our website as it breaks the system down into manageable components and means our database can be accessed from a range of locations. An advantage of the client server is that network peripherals, backups and network security are controlled centrally. Some of the cons of the client server are that is expensive to purchase and specialist staff such is needed (e.g. Network Manager). By using a Client/Server model we are best able to satisfy our non-functional requirements from our SRS, such as to enable the site to be accessible from multiple types of devices, so that the client is not restricted in the way they access the site.

From a high enough perspective (see 'Higher Level Model') a Layered Architectural pattern can be identified, however this is more useful in illustrating a general architecture of grouped together components rather than demonstrating how the system will work together as a whole. When considering the flow of data from: the web pages accessed by the user, to methods contained within files are hosted on a web server, which facilitate the transactions between the database servers storing information on products, user preferences and retailers; it can be seen how the design of our system naturally lends itself to a client server pattern as users will need a comprehensive, web-accessible interface which distributes data which is stored most effectively stored on in a database.

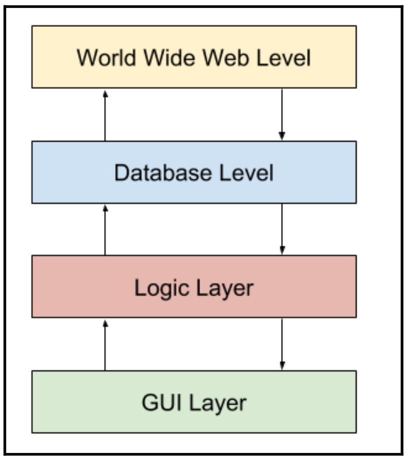
We also be used a bottom-up design approach. Although this requires more planning ahead than top-down design, it seems appropriate for this project because it allows group members to work on separate components of the final product sooner than they could otherwise. This will allow us to work more effectively because we will not need to wait for higher-level components to be completed before moving onto our designated areas.

Architectural Modelling

Lower Level Model

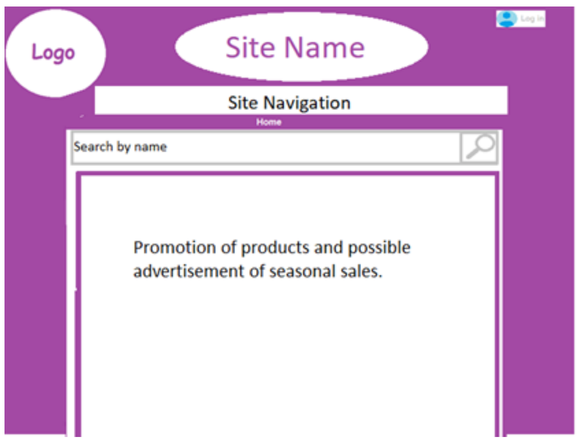


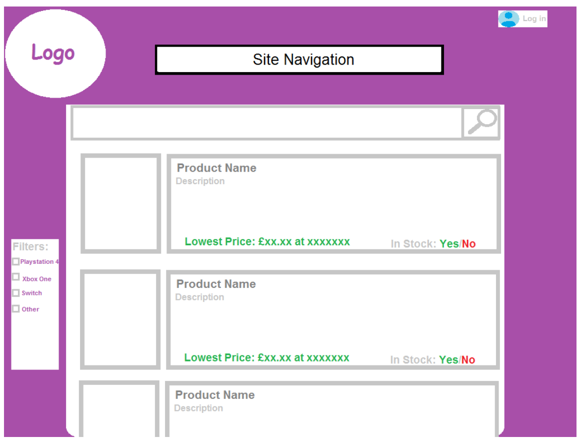
Higher Level Model



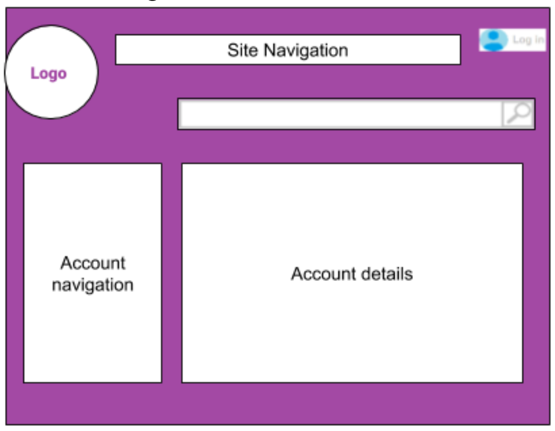
GUI Mockups

Home Page	Search Results

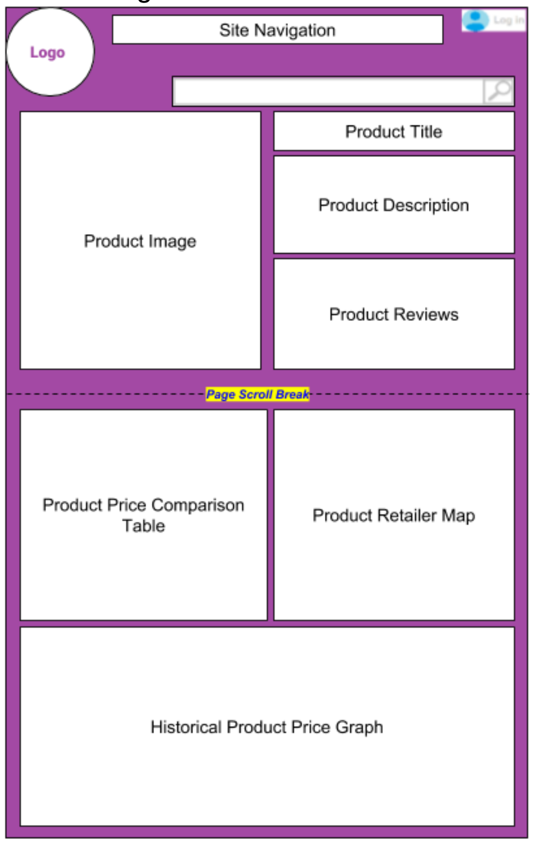

Logo
Site Name
Log in
Site Navigation
Home
Search by name
Promotion of products and possible advertisement of seasonal sales.



Account Page



Product Page



Appendix

Diagrams:
https://goo.gl/A2HhVX

Retrospective Report

What Worked

In terms of what aspects of the project were successful, we considered our Waterfall approach to development a success overall. It worked well due to our requirements being well defined in the system requirements submission, along with allowing each member to have clarity on exactly what point in the development process was currently underway. It also proved effective in minimising design errors, since all designs of the application had be finalised before a single line of code had been written.

Our teams communication was another strength of the project. We used various forms of communication throughout the process including Whatsapp, Skype and meetings on a weekly basis as a minimum. These all worked well since we could work separately whilst informing each other on our individual progress. The meetings were effective at allowing team members to give any feedback from tasks we had been working on, whilst giving an opportunity to assign more tasks for the next week to each team member.

At the start of the project, we allocated each member to sub teams that worked on different aspects of the design. This worked well for the most part since each member could focus on a specific area that they had prior experience on before the project such as GUI design.

What Didn't

One issue we faced is that not everyone in the group knew how to use key technologies such as git bash, which only came to our attention late into the project, during the implementation of the prototype. I think that if we had been thorough in identifying everyone's strong areas and gaps in knowledge, problems like this could have been avoided.

We also could have managed our time better. Generally, we started work on each stage of development as soon as the previous stage was submitted, but there may have been opportunities to do more by starting work on each stage sooner. Part of the reason this did not happen is because each phase of development was partially dependent on the ones that came before it (e.g. the design would determine how the prototype was built), but there were cases where we had enough information to start on the next part and did not.

The amount of work we could get done was also somewhat limited by the size of our team. We lost two of our members early on in development, which meant that the five people left had to take on a larger workload.

How Were the Risks Met

Several risks can be identified that could negatively impact the project. In order to mitigate the effects of these these risks, we needed to develop a methodology for dealing with potential problems that could arise, so that we could be confident in delivering the project on time and at a reasonable quality. One of these risks involves time constraints and deadlines. This was mitigated by following the Waterfall process model as it allowed us to easily structure the planning, requirements gathering, design, implementation, and testing stages of the project around the set deadlines. There was, however, a shortcoming in this decision as it left us with less time than we would have prefered for the implementation and testing stages of development.

Another risk we needed to overcome was the the effects of lacking personal expertise on critical technologies. We were able to overcome this, to an extent, by familiarising ourselves with the relevant skills before the implementation stage, so that we were better prepared to start implementation. As we had a significant amount of time from the project's initiation until implementation, this did not present much of a challenge and enabled most of the team members to develop most of the necessary skills.

A further risk we could have encountered would be that the project was not developed to the requirements of the users. We were able to prevent this not only through our requirements gathering techniques, but also through using the requirements we gathered from potential users as guidelines for development during the design and implementation stages, which enabled us to be sure that each design and implementation decision would we working towards satisfying user requirements.

Were the Estimates Correct

From the estimates view we had pretty close results, We weren't to achieve a few of the features because of time wise but we were able to achieve most off them. Some off the features we weren't able to conclude were (tips on when the best time is to buy a game, notifications of when a product has reduced in price, locations of the nearest retailers) During our work time we changed a few things on our work that we thought it will work better for the webapp.

The estimates we had for the architecture models we planned had distinct subsystems which would be the database, GUI and logic layer. The system architecture design closely follows our result relatively accurately, as the final system included separate components for the logic layer, GUI and database. Our GUI Mockups were close to the final design in terms of the color scheme used and the general layout.

Lessons Learned and Future Guidelines

From completing this coursework it’s clear to see that if the implementation had been started earlier our team would have been able to produce a much better application which encompassed all the features we initially wanted to implement such as a filter attached to the search bar and an embedded map showing all the locations of nearby retailers. Another lesson learnt was to better cater our project to team members’ expertise from the start rather than realising during the implementation that some team members were not suited to the task they had chosen. If we had realised this beforehand, tasks could be better assigned to match strengths.

Finally we learnt that the design our team had created could have been more specific rather than generalised because it would make it easier to plan out how the implementation section should fit together. This would also allow us to assign tasks to each team member weekly all related to a specific feature so that different features could be implemented quicker and more efficiently.