Activity B – OS – Checklist and Backup Sheet

Activity B PowerPoint

Activity B – Design Documentation

- ❑ *Worth 34 marks*

Activity B: The Design (Visual / Interface Design)

Your design must provide details of the solution to be implemented. It must be clear enough for a third party to use it to build the solution.

Include:

- ❑ *Layout and white space*
- ❑ *Visual hierarchies*
- ❑ *Common conventions*

Activity B: Data Requirements

Explain:

- ❑ *What data your system needs*
- ❑ *Where the data comes from*
- ❑ *How it will be stored*
- ❑ *How often it updates*

Activity B: Algorithm Design

You must show decomposition and algorithmic thinking.

Decomposition

- ❑ *Select key processes needed for the solution.*
- ❑ *Show decomposition through:*
- ❑ *Descriptions*
- ❑ *Decomposition diagrams*
- ❑ *Navigation maps*
- ❑ *Should fully cover:*
- ❑ *Inputs*
- ❑ *Processes*
- ❑ *Outputs*
- ❑ *Should be highly effective and comprehensive.*

Algorithms

- ❑ *May use:*
- ❑ *Flowcharts*
- ❑ *Pseudocode*
- ❑ *Data flow diagrams*
- ❑ *Static/dynamic model diagrams*
- ❑ *Or a combination*
- ❑ *Must produce consistently correct outcomes through:*
- ❑ *Precise logic*
- ❑ *Efficient structure and sequence*
- ❑ *Must use accepted conventions consistently.*

Activity B: Test Strategy

Explain how you will test your planned digital solution.

Your schedule must cover:

- ❑ *Understanding of how components interrelate*
- ❑ *The order components are tested*
- ❑ *Types of tests required (e.g., usability, unit tests, integration tests)*

Project Scenario: Student Operating System (OS)

Your team is designing a custom OS interface for college students (ages 16–21).

Purpose

- ❑ *Help students manage academic and personal tasks*
- ❑ *Provide easy access to apps, timetables, assignments, media tools, notifications*

Key Features (Examples)

- ❑ *Customisable homepage*
- ❑ *App dock*
- ❑ *Calendar integration*
- ❑ *Media player*
- ❑ *Note-taking widgets*

Complexity Requirements

You must demonstrate:

- ❑ *Front-end processes*
- ❑ *Back-end processes*
- ❑ *Interaction with data storage*
- ❑ *User profiles*
- ❑ *Real-time updates*

Programming Requirement

Show how the solution can be built using at least two languages, such as:

- ❑ *HTML/CSS (interface)*
- ❑ *Python or JavaScript (logic & data handling)*

Distinction Examples

DISTINCTION DESIGN DOCUMENT CHECKLIST

1. Visual / Interface Designs

Logo

- ❑ Logo created (e.g., using Canva)
- ❑ Inspiration for logo explained
- ❑ Justification of symbolism (e.g., tiger icon, brand meaning)
- ❑ Founding year included or placeholder used
- ❑ Explanation of inclusivity (suitable for children & adults)

Colour Scheme

- ❑ Industry/competitor research conducted
- ❑ Justification for chosen colours
- ❑ Reference to archetypes/psychology
- ❑ Palette created using Coolors
- ❑ Hex codes listed
- ❑ Palette shown as:
- ❑ List
- ❑ Array format
- ❑ Object/JSON format
- ❑ WCAG contrast checks performed
- ❑ Adjustments documented if colours failed contrast checks

Navigation Bar

- ❑ Fully designed navigation bar in UI
- ❑ Mention that tweaks may occur during development
- ❑ Notes on backend-informed adjustments (e.g., booking system)

UI Screen Designs (Figma)

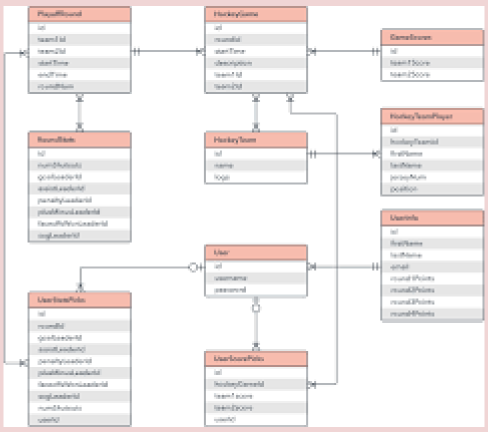
Include page designs for:

- ❑ Home Page
- ❑ About Page
- ❑ Educational Visits Page
- ❑ Booking System Page
- ❑ Login/Register Page
- ❑ Full Website Solutions Overview

2. Data Requirements

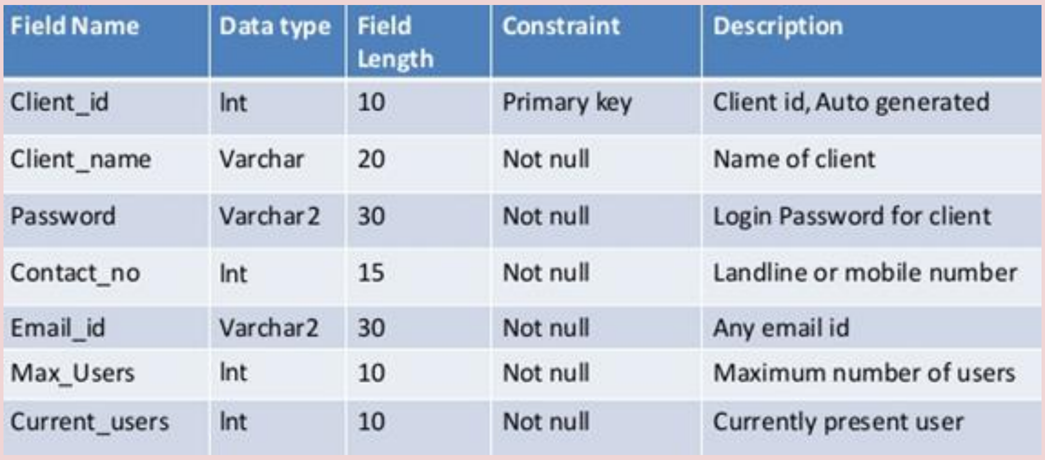
ERD + Data Flow

- ❑ Full ERD included
- ❑ Closer zoomed-in ERD version included



Data Dictionary

- ❑ Full data dictionary table included



Field Name	Data type	Field Length	Constraint	Description
Client_id	Int	10	Primary key	Client id, Auto generated
Client_name	Varchar	20	Not null	Name of client
Password	Varchar2	30	Not null	Login Password for client
Contact_no	Int	15	Not null	Landline or mobile number
Email_id	Varchar2	30	Not null	Any email id
Max_Users	Int	10	Not null	Maximum number of users
Current_users	Int	10	Not null	Currently present user

3. Algorithms & Flow

Wireframes

- ❑ Wireframes included
- ❑ Logical structure explained
- ❑ Flow description included

Booking System Flowcharts

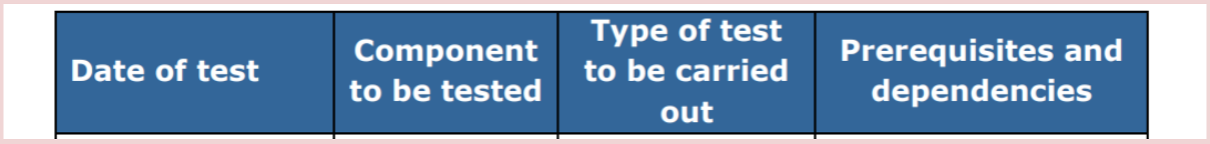
Four views must be included:

- ❑ Full flowchart
- ❑ Closer view – main logic
- ❑ Closer view – room booking logic
- ❑ Closer view – ticket booking logic

4. Testing Strategy

General -

Considering the scope of the project, we are going to be using manual testing. However, for future considerations, automated testing also can be used.

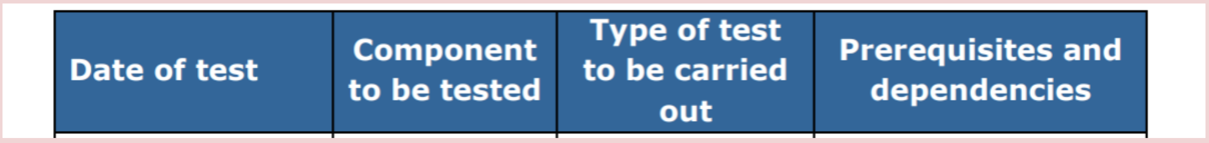


Date of test	Component to be tested	Type of test to be carried out	Prerequisites and dependencies

Considering the scope of the project, we are going to be using manual testing. However, for future considerations, automated testing also can be used.

- ❑ *Statement on choosing manual testing*
- ❑ *Mention future automated testing potential*

Test Table Requirements



Date of test	Component to be tested	Type of test to be carried out	Prerequisites and dependencies

For each component, include:

- ❑ *Date of test*
- ❑ *Component name*
- ❑ *Type of test*

TYPES

- ❑ *Black box*
- ❑ *White box*
- ❑ *Functional*
- ❑ *Integration*
- ❑ *Unit*
- ❑ *Non-functional (Compatibility)*

- ❑ *Prerequisites & dependencies listed*
- ❑ *Expected inputs (if any)*
- ❑ *Expected behaviour*

Components to Test

Ensure checklist includes tests for:

- ❑ *Navigation bar*
- ❑ *Homepage*
- ❑ *About page*

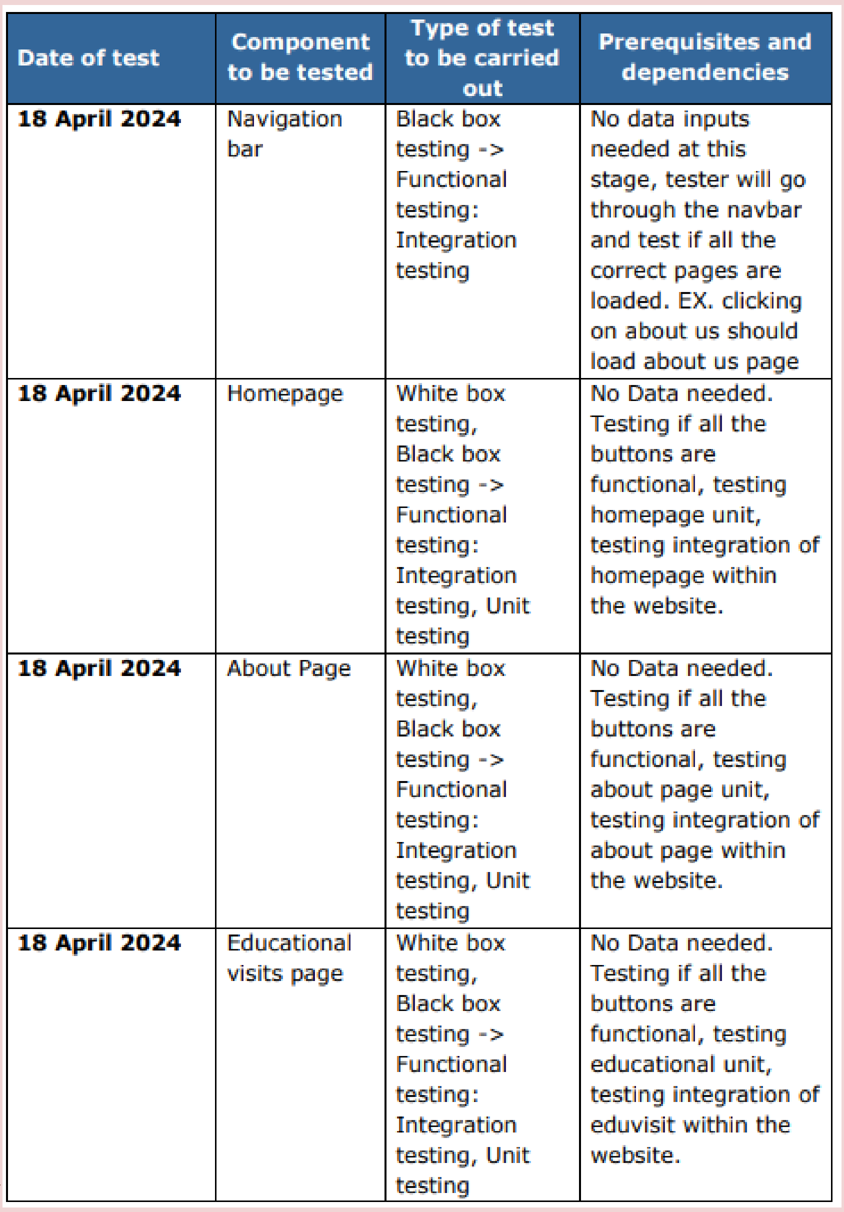
- ❑ *Educational visits page*
- ❑ *Booking page*
- ❑ *Login page*
- ❑ *Sign up page*
- ❑ *Backend booking system*
- ❑ *Backend login system*
- ❑ *Backend sign-up system*

Compatibility Testing

Test each page on:

- ❑ *iOS*
- ❑ *macOS*
- ❑ *Windows*
- ❑ *Android*

  Browsers:
- ❑ *Chrome*
- ❑ *Edge*
- ❑ *Firefox*
- ❑ *Opera*
- ❑ *Safari*

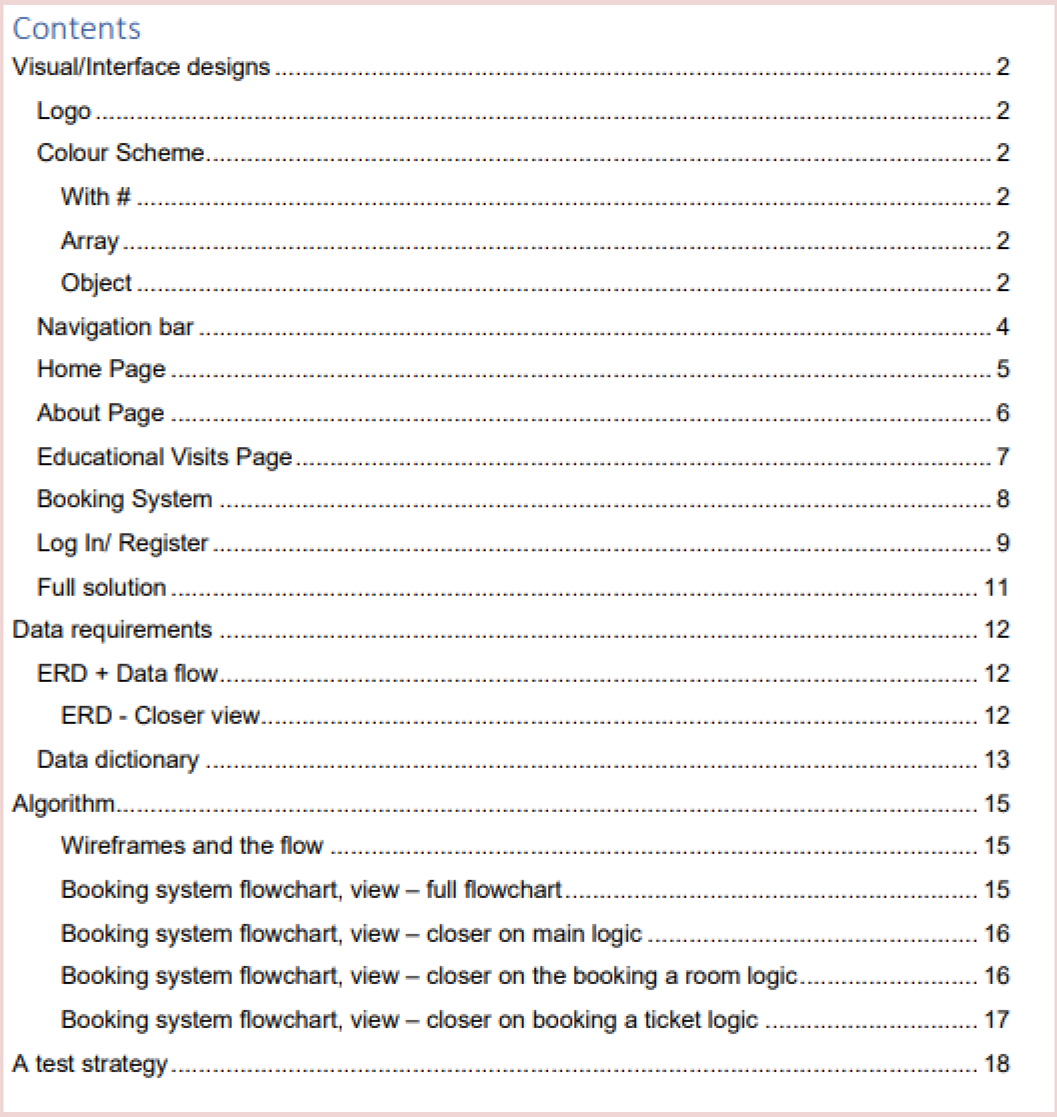


Date of test	Component to be tested	Type of test to be carried out	Prerequisites and dependencies
18 April 2024	Navigation bar	Black box testing -> Functional testing: Integration testing	No data inputs needed at this stage, tester will go through the navbar and test if all the correct pages are loaded. EX. clicking on about us should load about us page
18 April 2024	Homepage	White box testing, Black box testing -> Functional testing: Integration testing, Unit testing	No Data needed. Testing if all the buttons are functional, testing homepage unit, testing integration of homepage within the website.
18 April 2024	About Page	White box testing, Black box testing -> Functional testing: Integration testing, Unit testing	No Data needed. Testing if all the buttons are functional, testing about page unit, testing integration of about page within the website.
18 April 2024	Educational visits page	White box testing, Black box testing -> Functional testing: Integration testing, Unit testing	No Data needed. Testing if all the buttons are functional, testing educational unit, testing integration of eduvisit within the website.

EG

5. Final Section

- ❑ *Document labelled appropriately (e.g., T Level Digital, Distinction, Task 1 Design Docs)*



Contents

Visual/Interface designs 2
Logo 2
Colour Scheme 2
With # 2
Array 2
Object 2
Navigation bar 4
Home Page 5
About Page 6
Educational Visits Page 7
Booking System 8
Log In/ Register 9
Full solution 11
Data requirements 12
ERD + Data flow 12
ERD - Closer view 12
Data dictionary 13
Algorithm 15
Wireframes and the flow 15
Booking system flowchart, view – full flowchart 15
Booking system flowchart, view – closer on main logic 16
Booking system flowchart, view – closer on the booking a room logic 16
Booking system flowchart, view – closer on booking a ticket logic 17
A test strategy 18

Testing types and their definitions

Testing Type	Definition	Examples
Functionality Testing	Validates that the software features work according to requirements.	Checking login functionality; verifying form submissions; testing search results.

Usability Testing	Evaluates how easy and intuitive the system is for users.	Observing users navigating the UI; measuring time to complete tasks; collecting feedback on layout.
Interface Testing	Tests interactions between internal components or system interfaces.	Testing API responses; UI-to-database communication; verifying error messages.
Database Testing	Ensures data integrity, accuracy, and performance within the database.	Testing CRUD operations; verifying schema constraints; executing stored procedures.
Compatibility Testing	Confirms the software works across different environments.	Testing multiple browsers; OS versions; device sizes.
Performance Testing	Measures speed, stability, and scalability under load.	Load tests; stress tests; response time measurement.
Security Testing	Identifies vulnerabilities and ensures data protection.	Penetration testing; SQL injection checks; authentication/authorization testing.
Crowd Testing	Uses diverse external testers in real-world scenarios.	Public bug bounties; globally distributed testers trying the app.
Accessibility Testing	Ensures software is usable by people with disabilities.	Screen reader tests; colour contrast checks; keyboard-only navigation.
Acceptance Testing	Ensures the product meets business requirements and is ready for release.	Client UAT sessions; running real-world workflows against requirements.
Alpha Testing	Internal testing done before releasing to external users.	Testing early builds; identifying major bugs pre-beta.

Beta Testing	External testing by selected users in real-world conditions.	Early release for feedback; capturing usability issues.
Black Box Testing	Tests software without viewing internal code; focuses on inputs/outputs.	Testing form validation; verifying outputs; checking error handling.
White Box / Structural Testing	Tests using full knowledge of internal code, logic, and structure.	Unit tests; code coverage checks; path analysis.