

Table of Contents

[1.0 Introduction 3](#_Toc142905926)

[1.1 Background 3](#_Toc142905927)

[1.2 Scope 3](#_Toc142905928)

[1.3 Document contents 3](#_Toc142905929)

[2.0 Work Breakdown Structure 4](#_Toc142905930)

[3.0 Activity Definition & Estimation 7](#_Toc142905931)

[4.0 Gantt Chart 11](#_Toc142905932)

# Introduction

## Background

In the dynamic landscape of modern decision-making, data-driven insights have emerged as the linchpin of informed strategies. Our current initiative converges at the crossroads of data analysis and the vibrant Sydney Airbnb market, encapsulating the essence of a meticulously curated dataset sourced from Kaggle.

With a resolute goal, we aim to engineer a sophisticated software solution that unlocks the latent insights embedded within this dataset. It spans a gamut of attributes, spanning from property features to guest feedback, offering a panoramic view into the intricate fabric of the Sydney Airbnb ecosystem.

Beyond its observational role, our software encompasses a spectrum of functionalities tailored to empower users. From analysing suburb-specific trends over time to visualizing price dynamics, our software crystallizes the intricate dynamics of the Sydney Airbnb domain.

Moreover, the software delves into guest sentiments, deciphering the nuances of customer comments—a pivotal factor in shaping guest experiences. Infused with innovation, the software introduces an element of novelty, elevating the overall analytical experience.

At its core, this initiative rests upon the foundation of a meticulously curated Kaggle dataset—a beacon of real-world data. It's a pivotal bedrock upon which our software thrives, unfurling a world of possibilities.

Beyond abstract data, this endeavour holds the potential to optimize rentals, enhance guest experiences, and empower property owners with actionable insights. As we embark on this data-driven journey, the Sydney Airbnb market beckons—an intricate narrative waiting to be decoded. Our software, an embodiment of this vision, stands ready to reshape how we navigate the realm of transient lodging.

## Scope

The scope of this project is to design and implement a sophisticated software application for analysing the dynamic Sydney Airbnb market. The software will offer users the ability to interact with a curated dataset, enabling data retrieval, visualization, and insights generation. Key functionalities include querying property information based on user-defined criteria, creating informative price distribution charts, conducting keyword-based searches, and analysing customer feedback on cleanliness.

Beyond the technical development, the project also encompasses essential documentation phases. A comprehensive project plan will outline the project's lifecycle from inception to deployment, ensuring seamless coordination and progress tracking. The software design document will detail the architecture, user interfaces, and system components, aligning development with user-cantered design principles.

The project's ultimate ambition reaches beyond code to cultivate an enriching data ecosystem. By empowering users with tools for informed decision-making, property owners for optimizing rentals, and data enthusiasts for exploration, this Endeavor aims to amplify the value of data analysis in the context of the vibrant Sydney Airbnb market.

In essence, the project's scope encompasses technical excellence, strategic planning, and the creation of an impactful platform that transforms raw data into actionable insights, driving informed choices and enhancing understanding of the Airbnb market in Sydney.

## Document contents

This section outlines the specific contents and structure of the document. It provides readers with an overview of what to expect in the following sections:

* Work Breakdown Structure (WBS):

This section presents a detailed breakdown of the project's tasks and subtasks. It outlines the hierarchical structure of the project, starting from the initial tasks and branching out into smaller components. Each task is assigned a unique ID, and its description, duration, and any dependencies are provided. The WBS offers a comprehensive view of the project's scope and helps in understanding the sequence of tasks and their relationships.

* Activity Definition & Estimation:

In this section, each activity identified in the WBS is defined in more detail. The purpose, specific steps, and estimated time required for each activity are elaborated upon. By breaking down the tasks further, the project team gains a clearer understanding of the work involved and the effort needed to complete each activity successfully.

* Gantt Chart:

The Gantt chart, included in this section, visually represents the project timeline. It plots the start and end dates of each task on a timeline, showcasing their dependencies and durations. The Gantt chart provides a clear visualization of how tasks are scheduled and when they are expected to be completed. This aids in tracking progress and managing resources effectively throughout the project.

By structuring the document with these sections, readers gain insights into the project's structure, its component tasks, their estimated durations, and the overall project timeline. This content ensures that stakeholders understand what the document contains and can navigate through the information effectively.

# Work Breakdown Structure

The Work Breakdown Structure (WBS) section serves as a comprehensive outline of the project's tasks and activities, offering a high-level overview of the project's scope and components. It presents a hierarchical breakdown of the entire project into manageable work units, each representing a distinct task or activity. The WBS provides a structured framework to capture all essential elements of the project, guiding the subsequent activities and aiding in effective project management.

In this section, the WBS is presented in the form of a table, with each row corresponding to a specific task or activity. The tasks are organized in a logical sequence, often categorized by project phases or themes. While the WBS focuses on outlining the scope of the project, it forms the foundation for generating the Activity Definition and subsequently scheduling tasks in the Gantt Chart. The WBS provides a visual roadmap that helps stakeholders, team members, and project managers gain a clear understanding of the project's composition and the sequence of tasks required for successful completion.

|  |  |  |  |
| --- | --- | --- | --- |
| **Work Breakdown Structure** | | | |
| **Task ID** | **Task Description** | **Duration** | **Predecessor** |
| **Project Initiation** | | | |
| 1 | Gather group members | 1 Day | 0 |
| 2 | Select dataset | 1 Day | 1 |
| 3 | Set up GitHub repository | 1 Day | 2 |
| **Preparatory Work** | | | |
| 4 | Understand dataset structure | 2 Days | 2 |
| 5 | Identify analysis tasks | 1 Day | 4 |
| 6 | Define additional analysis task | 1 Day | 5 |
| 7 | Discuss software design approach | 2 Days | 5 |
| **Prepare Project Plan** | | | |
| 8 | Create Project Overview | 1 Day | 4 |
| 9 | Develop Work-Breakdown Structure | 2 Days | 5 |
| 10 | Define Activity Definitions and estimates | 1 Day | 9 |
| 11 | Build Gantt chart | 2 Days | 10 |
| **Prepare Software Design Document** | | | |
| 12 | Create System Vision Statement | 1 Day | - |
| 13 | Identify and list user requirements | 1 Day | 12 |
| 14 | Identify and list software requirements | 1 Day | 12 |
| 15 | Define use cases and personas | 2 Days | 13,14 |
| 16 | List system components and design | 3 Days | 15 |
| **UI Wireframe Design** | | | |
| 17 | Design early user interface wireframe/mock-ups | 2 Days | 16 |
| 18 | Refine and finalize wireframe design | 1 Day | 17 |
| **Software Implementation and Testing** | | | |
| 19 | Implement Data Analysis Program |  |  |
| 19.1 | Set up project structure and environment | 1 Day | 18 |
| 19.2 | Data Analysis and insights | 2 Days | 19.1 |
| 19.3 | Develop user interface components | 3 Days | 19.2 |
| 19.4 | Implement data analysis algorithms with UI | 4 Days | 19.3 |
| 19.5 | Integrate 5 types of insight tools | 2 Days | 19.4 |
| 19.6 | Data visualization | 1 Day | 19.5 |
| 20 | Perform Unit Testing |  |  |
| 20.1 | Create test cases for data loading | 1 Day | 19.6 |
| 20.2 | Execute unit tests for UI components | 2 Days | 20 |
| 20.3 | Conduct unit tests for analysis | 3 Days | 20.1 |
| 20.4 | verify UI and analysis integration | 2 Days | 20.2 |
| 20.5 | Perform unit testing and debugging | 4 Days | 20.3 |
| **Software Testing Report** | | | |
| 21 | Compile test cases and outcomes | 3 Day | 20.4 |
| 22 | Include coverage and evaluation details | 1 Day | 21 |
| 23 | Document acceptance testing results | 2 Day | 22 |
| 24 | Explain implementation and testing process | 1 Day | 23 |
| **Executive Summary** | | | |
| 25 | Summarize project objectives and outcomes | 7 Days | 24 |
| 26 | Highlight key achievements and results | 5 Days | 24 |
| 27 | Include main findings and recommendations | 3 Days | 24 |
| 28 | Edit and proofread summary | 1 Day | 27 |
| **Final Editing and Review** | | | |
| 29 | Review Project plan, testing and summary reports | 7 & 5 Days | 28 |
| 30 | Revise and edit content | 3 & 3 Days | 29 |
| 31 | Track actual completion times | 1 & 2 Days | 28 |
| 32 | Regularly update Gantt chart | 6 & 7 Days | 28 |
| 33 | Git log text document | 1 & 3 Days | 28 |
| 34 | Finalize all documentation | 2 & 2 Days | 33 |
| **Final Review and submission** | | | |
| 35 | Revise and finalize everything | 2 & 2 Days | 34 |
| 36 | Submit all project deliverables | 1 & 1 Day | 35 |

# Activity Definition & Estimation

The section on "Activity Definition & Estimation" serves as a pivotal guidepost for steering the project from inception to completion. Within this segment, each project task finds its distinct identity as activities are meticulously defined and estimated. These activities span the entire project lifecycle, from initial project initiation and preparatory work to software design, implementation, testing, and finalization. Each activity is thoughtfully numbered and aligned with the project's timeline, reflecting its position in the comprehensive Gantt chart. The activities are accompanied by concise yet comprehensive descriptions, offering insight into their objectives, prerequisites, and expected outcomes. Alongside this, the estimations of time required for each task serve as a practical resource for constructing an accurate project timeline. This section, in essence, lays out the groundwork for efficient project management, ensuring a structured and well-defined path towards achieving the project's goals.

Project Initiation:

1. Gather Group Members (1 Day):

* Contact potential team members, discussing their skills and availability.
* Formally invite selected members to join the project team.

1. Select Dataset (1 Day):

* Research available Airbnb datasets on platforms like Kaggle and data repositories.
* Evaluate dataset options based on size, relevance, and data quality.
* Choose the most suitable dataset for the project's analysis goals.

1. Set up GitHub Repository (1 Day):

* Create a new GitHub repository for the project.
* Configure repository settings, including access permissions and collaboration guidelines.
* Initialize the repository with essential files, such as a README and license.

1. Preparatory Work: 4. Understand Dataset Structure (2 Days):

* Download and explore the selected dataset's contents.
* Identify the various data fields, their types, and potential relationships.
* Document an overview of the dataset's structure for reference.

1. Identify Analysis Tasks (1 Day):

* Brainstorm potential analyses to extract insights from the dataset.
* Prioritize analysis tasks based on their relevance and potential impact.
* Document a list of analysis objectives and associated tasks.

1. Define Additional Analysis Task (1 Day):

* Collaboratively brainstorm to identify an extra analysis task beyond the initial list.
* Evaluate the feasibility and significance of the additional task.
* Document the details and purpose of the new analysis task.

1. Discuss Software Design Approach (1 Day):

* Conduct a team meeting to discuss software design considerations.
* Analyse various architectural approaches and their pros and cons.
* Choose a suitable design approach based on the project's objectives.

Prepare Project Plan:

1. Create Project Overview (1 Day):

* Write a concise overview of the project, including its goals, scope, and significance.
* Summarize the dataset's relevance and potential impact on the software.
* Provide an outline of the document's contents.

1. Develop Work-Breakdown Structure (1 Day):

* Break down the project into smaller tasks and sub-tasks.
* Organize these tasks in a hierarchical structure, showing their relationships.
* Ensure that each task is well-defined and contributes to the project's objectives.

1. Define Activity Definitions and Estimates (1 Day):

* Elaborate on each task in the work breakdown structure.
* Specify the purpose, expected outcomes, and deliverables of each task.
* Estimate the time required for each task based on its complexity and dependencies.

1. Build Gantt Chart (2 Days):

* Translate the activity definitions into a visual timeline.
* Establish task dependencies to create a sequential order of execution.
* Assign estimated durations to each task and generate a Gantt chart.

Prepare Software Design Document:

1. Create System Vision Statement (1 Day):

* Define the purpose and objectives of the software.
* Describe the problem domain and how the software addresses it.
* Highlight the potential benefits and impacts of the software.

1. Identify and List User Requirements (1 Day):

* Gather requirements by engaging potential users or stakeholders.
* List the features, functionalities, and interactions expected from the software.
* Ensure that the requirements are clear, concise, and aligned with user needs.

1. Identify and List Software Requirements (2 Days):

* Derive technical requirements from the user requirements.
* Detail the software functionalities, data handling, and processing capabilities.
* Ensure that the software requirements are complete and unambiguous.

1. Define Use Cases and Personas (2 Days):

* Describe the scenarios in which users will interact with the software.
* Create use case diagrams to visualize user interactions and system responses.
* Develop user personas to better understand the software's target audience.

1. List System Components and Design (3 Days):

* Identify the major components/modules of the software.
* Describe the purpose and responsibilities of each component.
* Sketch a high-level design or flowchart depicting the interactions between components.

UI Wireframe Design:

1. Design Early User Interface Wireframe/Mock-ups (2 Days):

* Create initial wireframes illustrating the layout and structure of the user interface.
* Focus on representing key elements and interactions without detailed visuals.
* Gather feedback from team members to refine the wireframes.

1. Refine and Finalize Wireframe Design (1 Day):

* Incorporate feedback and adjustments into the wireframes.
* Ensure that the wireframes accurately reflect the intended user interface.
* Finalize the wireframes for further design and implementation.

Software Implementation and Testing:

19. Implement Data Analysis Program:

19.1 Set up project structure and environment (1 Day):

* Create the directory structure for the project.
* Set up the development environment, including programming languages and tools.
* Configure project-specific settings.

19.2 Data Analysis and insights (2 Days):

* Load and preprocess the selected dataset.
* Explore the data to gain insights and identify patterns.
* Perform statistical analyses to extract meaningful information.

19.3 Develop user interface components (3 Days):

* Design and create graphical user interface (GUI) elements.
* Implement interactive features for data input and visualization.
* Ensure the user interface is intuitive and user-friendly.

19.4 Implement data analysis algorithms with UI (4 Days):

* Develop algorithms to perform specific analyses on the dataset.
* Integrate analysis algorithms with the user interface.
* Ensure data flows seamlessly between the UI and analysis modules.

19.5 Integrate 5 types of insight tools (2 Days):

* Identify and implement five distinct insight tools or features based on analysis results.
* Ensure each insight tool provides valuable information to users.
* Integrate these tools into the user interface for easy access.

19.6 Data visualization (1 Day):

* Create visualizations (charts, graphs, etc.) to represent analysis results.
* Ensure visualizations effectively communicate insights.
* Implement interactive features for users to interact with visualized data.

20. Perform Unit Testing:

20.1 Create test cases for data loading (1 Day):

* Define test scenarios to verify data loading accuracy.
* Develop test datasets to cover different scenarios.
* Create test scripts to automate the testing process.

20.2 Execute unit tests for UI components (2 Days):

* Design and implement unit tests for user interface features.
* Validate that user interactions produce the expected outcomes.
* Automate UI testing where possible.

20.3 Conduct unit tests for analysis (3 Days):

* Create comprehensive test cases to validate analysis algorithms' correctness.
* Verify that algorithms produce accurate results for various data inputs.
* Automate testing of analysis algorithms.

20.4 Verify UI and analysis integration (2 Days):

* Test the interaction between UI components and analysis modules.
* Ensure data flows correctly between different parts of the software.
* Validate that insight tools are seamlessly integrated.

20.5 Perform unit testing and debugging (4 Days):

* Execute comprehensive unit tests for the entire software.
* Identify and fix any bugs or issues encountered during testing.
* Debug and optimize the software for performance and reliability.

Software Testing Report:

21. Compile test cases and outcomes (1 Day):

* Gather all test cases and their expected outcomes.
* Document the test scenarios and expected results for each module.

22. Include coverage and evaluation details (1 Day):

* Present the coverage achieved by the unit tests.
* Evaluate the effectiveness of the unit testing approach.

23. Document acceptance testing results (1 Day):

* Detail the results of acceptance testing conducted by stakeholders.
* Record any feedback or issues encountered during acceptance testing.

24. Explain implementation and testing process (1 Day):

* Describe the implementation process followed for the software.
* Explain the approach used for unit testing, integration testing, and acceptance testing.

Executive Summary:

25. Summarize project objectives and outcomes (1 Day):

* Provide a concise overview of the project's goals and intentions.
* Highlight the desired outcomes and benefits of the completed software.

26. Highlight key achievements and results (1 Day):

* Showcase the significant accomplishments achieved throughout the project.
* Emphasize milestones, breakthroughs, and notable contributions.

27. Include main findings and recommendations (1 Day):

* Present the key insights and discoveries derived from data analysis.
* Offer recommendations based on the analysis results for informed decision-making.

28. Edit and proofread summary (1 Day):

* Review and refine the executive summary's content for clarity and coherence.
* Ensure that the language is polished and effectively communicates the project's essence.

Final Review and Submission:

29. Review Project plan, testing, and summary reports (17 Days):

* Thoroughly review the project plan, testing reports, and executive summary.
* Identify any discrepancies, errors, or areas that require improvement.

30. Revise and edit content (6 Days):

* Address any identified issues and make necessary revisions.
* Ensure consistency and accuracy in the documentation's content.

31. Track actual completion times (3 Days):

* Compare the estimated durations with the actual time taken for each task.
* Analyze any deviations and update project documentation accordingly.

32. Regularly update Gantt chart (1 Day):

* Update the Gantt chart with the actual completion times of tasks.
* Reflect any changes in task dependencies or durations.

33. Git log text document (10 Days):

* Create a log documenting significant changes made to the project's codebase.
* Include information about commits, branches, and collaborators.

34. Finalize all documentation (5 Days):

* Review all project documentation for completeness and accuracy.
* Ensure that the documentation adheres to established formatting and style guidelines.

Final Review and Submission (Continued):

35. Revise and finalize everything (2 Days):

* Conduct a final review of all project deliverables and documentation.
* Make any last-minute adjustments or refinements to ensure quality.

36. Submit all project deliverables (1 Day):

* Package and organize all project documentation, code, and assets.
* Submit the final deliverables to the relevant stakeholders or platforms.

# Gantt Chart

The Gantt chart presented in this document acts as a visual compass, guiding the project through its various stages with a clear and organized timeline. Each task outlined in the activity definition section finds its place on the Gantt chart, presenting a dynamic representation of the project's progression. The Gantt chart offers a holistic view of task interdependencies, helping project managers and stakeholders grasp the flow and sequencing of activities. It serves as a strategic planning tool, enabling efficient resource allocation, timeline management, and task prioritization. With each task graphically displayed along the timeline, the Gantt chart provides a tangible sense of project milestones, deadlines, and overall progress. This section underscores the Gantt chart's significance as an instrumental tool in project monitoring, fostering effective coordination, and facilitating informed decision-making throughout the project lifecycle.

*A screenshot of a computer

Description automatically generated*