**Project Plan**

**AJJ BNB**

**Student Names:**

**Ashley Pergoliti (s5311775)**

**Jonas Sajonas (s5284977)**

Table of Contents

[1.0 Introduction 3](#_Toc147487333)

[1.1 Background 3](#_Toc147487334)

[1.2 Scope 3](#_Toc147487335)

[1.3 Document contents 4](#_Toc147487336)

[2.0 Work Breakdown Structure 5](#_Toc147487337)

[3.0 Activity Definition & Estimation 6](#_Toc147487338)

[4.0 Gantt Chart 10](#_Toc147487339)

# Introduction

The project proposal to AJJ BNB is contained within this document and the accompanying software design document. The proposal is to create a software solution that extracts and analyses the public Sydney Airbnb data set hosted on kaggle.com. The solution will enable the company to become data-driven and make informed insights and decisions in the Airbnb spaces and trends to compete in a challenging market.

## Background

In today's data-driven business landscape, utilising gathered data means gaining the  
edge in the competition with rivals. AJJ BNB is at the forefront of the accommodation rivalry in Sydney, they will benefit the most by staying ahead of the competition. Sydney is a sought-after destination for travellers, tourists, and businesses alike. At the same time, Airbnb is a popular choice for accommodation and a lucrative investment choice for property owners. Kaggle.com provides access to Sydney Airbnb Open Data. The data provided includes detailed listings, pricing, calendar variances, reviews, and more.

The proposed software aims to provide a user-friendly, quick access point to specific information and trends about the Sydney Airbnb market. Users can receive charted or listed data tailored to their specific requirements in as few steps as possible.

## Scope

The scope of this document covers the project management aspect of the proposed project. The supporting documentation included has been developed to plan out and structure the project tasks as well as allocate time estimates required to produce individual deliverables as well as the duration of the entire project. The project management documents are presented as deliverables themselves and are then used to staff and fund the project as well as provide phase timing to the client.

The included Work Breakdown Structure covers all the deliverables for the project and lays them out in sections including project management, software design, software implementation, testing and executive summaries. Each Section has a group of sub sections shown beneath.

The activity definition uses the deliverable structure from the WBS to provide a time estimate for each task. This is then used to create the Gantt chart

The Gantt chart shows a visual representation of the project’s timeline. The time estimates for each task are highlighted as well as showing where tasks are occurring concurrently. The Gantt chart shows the overall time commitments as well as the time required to perform each part. The Gantt chart allows us the find the critical path to the project, identifying which tasks must be completed before the others can commence.

## Document contents

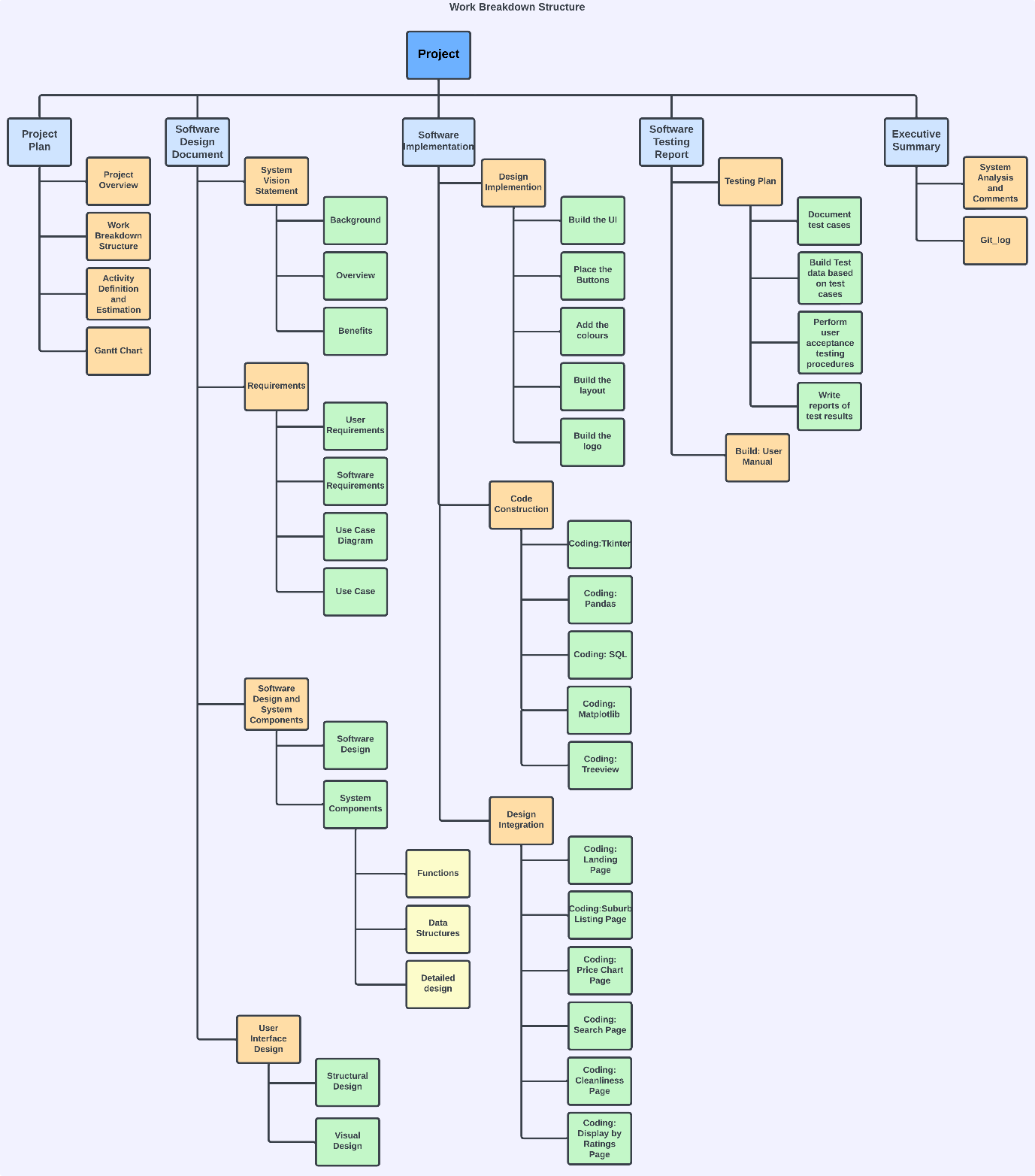
This document serves as an overview of the project plan. The Software Design Document contains detailed software information that thoroughly examines the software solution. A higher-resolution Gantt chart is also available for further clarity.

**Contents of this Document:**

* Background
* Scope
* Work Breakdown Structure
* Activity Definition and Estimation
* Gantt Chart

# 2.0 Work Breakdown Structure

This section uses a deliverable-phased approach to show the Work Breakdown Structure (WBS). It visualizes the project's scope, tasks, and progress for each phase and its unique deliverables. Each level 1 element in this structure represents the top-level definition, while level 2 elements represent individual deliverables.

Figure 1: Update Work Breakdown Structure

**Update**: The project was completed in the order the team had planned, with some minor changes that the team had decided to make. We chose Tkinter over Wxpython as our coding base for the layout, and we used Treeview to display tables specified in the developed use case.

# 3.0 Activity Definition & Estimation

This section presents a detailed breakdown of project activities derived from the phases outlined in the Work Breakdown Structure (WBS). Each activity is assigned a unique identifier, an activity name, comprehensive details, and an estimated time required for its completion.

|  |  |  |  |
| --- | --- | --- | --- |
| **Gantt Number** | **Item** | **Activity Details** | **Time Estimation** |
| **1** | **Project Plan** |  |  |
| 1.1 | Project Overview | Describe the overview of this project, including outlining the background, scope, and contents of the project plan document. | 1 days |
| 1.2 | Work Breakdown Structure | Create the Work Breakdown Structure (WBS), outlining the tasks and phases of the project. | 2 days |
| 1.3 | Activity Definition and Estimation | Describe the activities and duration estimates, along with a short description of each activity. | 2 days |
| 1.4 | Gantt Chart | Display the schedule and progress of each task, including the start, and end times of each task. | 2 days |
| **2** | **Software Design Document** |  |  |
| **2.1** | **System Vision Statement** | Describe the problem background, the overview of the new system, and the potential benefits of this system. |  |
| 2.1.1 | Background | Describe the background of the problems that users face. This outlines why the system is being created. | 1 day |
| 2.1.2 | Overview | Describes the tasks that the system should be able to do. | 1 day |
| 2.1.3 | Benefits | Describes the benefits of the software to the AJJ company team. | 1 day |
| **2.2** | **Requirements** |  |  |
| 2.2.1 | User Requirements | Description of what the user should be able to complete with the system | 1 day |
| 2.2.2 | Software Requirements | Description of what the software itself should be able to complete. | 1 day |
| 2.2.3 | Use Case Diagram | Visual representation of interactions between actors and system, based off the use cases. | 1 day |
| 2.2.4 | Use Cases | Description of how users interact with the system, with the flow of events, and alternative flow. | 1 day |
| **2.3** | **Software Design and System Components** | Planning for the Software Design and System Components |  |
| 2.3.1 | **Software Design** | Diagram/hierarchy that represents the flow of the system. Including system actions, and user actions and choices, from start to stop. | 1 day |
| **2.3.2** | **System Components** | The design of the different aspects of the app, including functions, data structures, and the pseudocode of the algorithms/functions. |  |
| 2.3.2.1 | Functions | A table representing the functions of the system, including the function name, description of the function, parameters used in the function, side effects of the function, and the return value of the function. | 4 days |
| 2.3.2.2 | Data Structures | Describes the data types and structures that will be used in this system, and the functions that will use these types. | 2 days |
| 2.3.2.3 | Detailed design | Pseudocode created for a few functions and algorithms that will be used in this system. | 3 days |
| **2.4** | **User Interface Design** | Planning for the User Interface Design |  |
| 2.4.1 | Structural Design | The hierarchy of the system, which outlines the different pages of the system, and how they interact during system use. | 3 days |
| 2.4.2 | Visual Design | Outlines the design of the system, including a colour palette and wireframes. | 3 days |
| **3** | **Software Implementation** |  |  |
| **3.1** | **Design Implementation** | Creation of the System Components. |  |
| 3.1.1 | Build the User Interface | Build the User Interface. | 3 days |
| 3.1.2 | Place the Buttons | Place buttons on the User Interface. | 1 day |
| 3.1.3 | Add the Colours | Add Colours to the User interface and make sure of the functionality. | 3 days |
| 3.1.4 | Build the Layout | Build the System Layout and Structure. | 5 days |
| 3.1.5 | Build the Logo | Design and Create the Company Brand. | 2 days |
| **3.2** | **Code Construction** | The Creation of all the core components for the system. |  |
| 3.2.1 | Coding: Tkinter | The code for creating the graphs and charts. | 3 days |
| 3.2.2 | Coding: Pandas | The code for creating the UI design of the app. | 3 days |
| 3.2.3 | Coding: Matplotlib | The code for interacting with Excel documents or files. | 3 days |
| 3.2.4 | Coding: Treeview | The code for creating the graphs and charts. | 3 days |
| **3.3** | **Design Integration** | Process of combining the different components of the system. |  |
| 3.3.1 | Coding: Landing Page | Integrate the Landing Page design with the code and functionality. | 1 day |
| 3.3.2 | Coding: Suburb Listing Page | Integrate the Suburb Listing Page design with the code and functionality. | 1 day |
| 3.3.3 | Coding: Price Chart Page | Integrate the Price Chart Page design with the code and functionality. | 1 day |
| 3.3.4 | Coding: Search Page | Integrate the Search Page design with the code and functionality. | 1 day |
| 3.3.5 | Coding: Cleanliness Page | Integrate the Cleanliness Page design with the code and functionality. | 1 day |
| 3.3.6 | Coding: Display by Ratings Page | Integrate the Display by Ratings Page design with the code and functionality. | 1 day |
| **4** | **Software Testing Report** |  |  |
| **4.1** | **Testing Plan** | Outlines the overall strategy, approach and objectives for testing a software system. |  |
| 4.1.1 | Document Test Cases | Create instructions on how a particular functionality the system should be tested. | 5 days |
| 4.1.2 | Build Test Data based on test Cases | Create the specific data conditions that will be used to execute and evaluate the outcomes of the Software test cases. | 1 day |
| 4.1.3 | Perform user acceptance testing procedures | Check if the Software Application meets the requirements of the intended Users. | 2 days |
| 4.1.4 | Write Reports of Test Results | Compile a summary of test results. | 1 day |
| **4.2** | **Build: User Manual** | Create the User Manual. | 1 day |
| **5** | **Executive Summary** |  |  |
| 5.1 | Analysis and Comments | Document Project Report and Analysis and Comments for the system. | 1day |
| 5.2 | Git\_log | Print the Summary of Git Log. | 1hr |

Table 1: Activity Definition and Estimation

**Update**: The activity definition and estimation have been updated to reflect our changes. Tkinter was used to structure the layout code, followed by Pandas code for the CSV reading. We wrote the SQL code to create the database and retrieve the relevant data. To display data, use that code with Matplotlib and Treeview. These were the only changes we made in total.

# 4.0 Gantt Chart

This section is dedicated to the Gantt Chart, a visual tool that depicts the timeline, tasks, dependencies, and progress of the project. It is a resource management and allocation tracker and a project scheduling tool. It also gives insights into project phases needing more attention and time allocation. The Critical Path is also defined, which is a component that tracks the longest series of tasks within the project. As each delay in activities along this path immediately extends the project's timeframe, the critical path represents the most significant period necessary for project completion. A low resolution of the Gantt chart is presented below, and a high resolution is available in its own document.

The Critical Path is delineated through the following sequence of tasks:

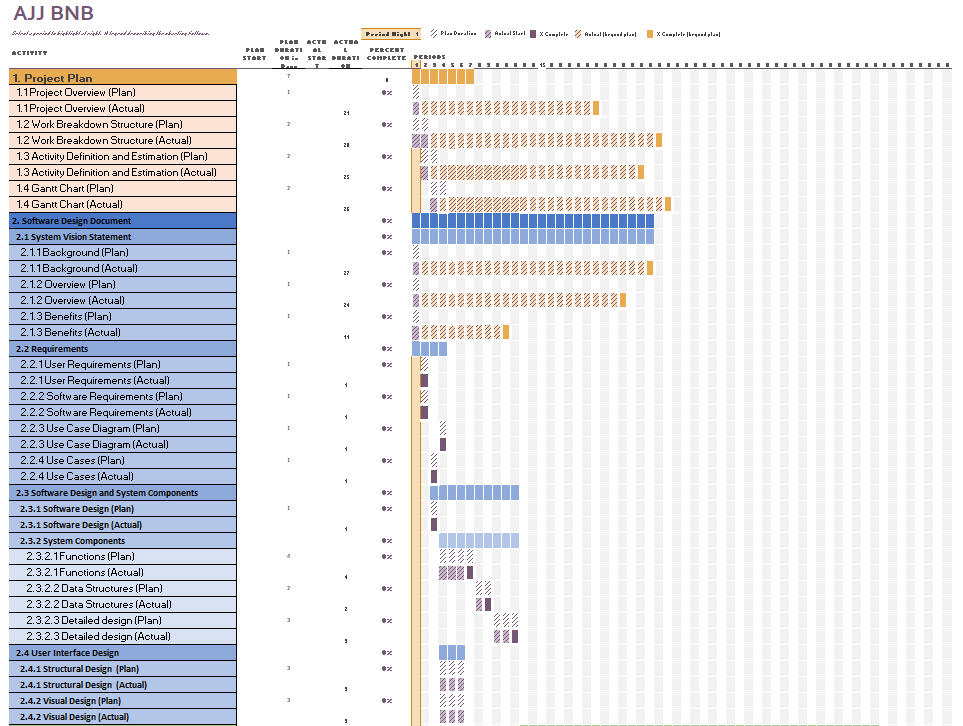
Project Overview > Activity Definition and Estimation > Functions > Data Structures> Detailed Design > Build Layout > Coding: Tkinter > Coding: Matplotlib > Document Test Cases> Build Test Data Based on Test Cases > Perform User Acceptance Testing Procedures > Write Reports of Test Results > Analysis and Comments.

Total Duration: 33 Days

Shown in the Gantt chart are the tasks completed to date. The actual time taken for the Project Plan and Software Design documents has far exceeded the estimates. This is due to the fact that we are working on the project part-time over a period of a few weeks. The time estimates were made with a real client in mind and assumed tasks completed by team members working full time. We have included our actual time as the total time of days passed from start to finish that we worked on the tasks.

**Update**: The Gantt chart was updated to reflect our changes, and the actual implementation was also added. We added three days for the Treeview code, but we could simultaneously be done with the other codes. The Critical Path stayed the same and unaffected.

**Gantt Chart**



A screenshot of a computer

Description automatically generated