Suite Dreams

Project Plan

Kyren Stephenson, Jonathan Fasano, Alexes Pagan, John Doyle, Arman Hemmati Alamdari, Parsa Jafaripour

# Contents

## 1. Introduction

1.1 Project Purpose……………………………………………………. 3

## 2. Project Overview

2.1 Project Deliverable………………………………………………... 3

2.2 Project Milestones………………………………………………… 3

2.3 Branching Policies………………………………………………… 4

2.4 Project Management and Control…………………………………. 4

## 3. Development Requirements

3.1 Hardware Requirements…………..……………………………… 5

3.2 Software Requirements…………………………………………… 5

## 4. Task Overview

4.1 Task Table……….………………………………………………… 6

## 5. Project Schedule

5.1 Gantt Chart…….…………………………………………………... 7

5.2 PERT Chart……….……………………………………………….. 7

## 6. Risk

6.1 Technical Risks…………………….……………………………… 8

6.2 Project Management Risks……………………………………….. 10

6.3 Compliance Risks………………………………………………… 11

6.4 External Risks…………………………………………………….. 12

6.5 Project Management and Control…………………………………. 13

## 7. Quality Assurance

7.1 Testing……….…………………………………………………….. 14

## 1. Introduction

1.1 Purpose

The purpose of this project plan is to provide a cohesive schedule and framework for the development of Suite Dreams. It intends to ensure a smooth and efficient process by identifying and understanding potential risks, time frame for development, and clearly defining the division of labor. This document includes visual tools such as a Gantt and PERT chart which both map the development process as well as show the critical path and slack between tasks. The process concludes with a successfully developed dormitory allocation software.

## 2. Project Overview and Organization

2.1 Project Deliverables

Suite Dreams version 1.0 will be completed and released after integration testing by Tuesday, December 3rd, 2024.

2.2 Project Milestones

* Students fill out a roommate questionnaire and answer questions regarding their preferences and habits. On this same document, they provide their room selections. Admins can search for and have access to student responses.
  + Each student’s responses are reviewed alongside another to provide a similarity score.
* With the pairing information and similarity scores, the program uses an algorithm to group the students into specific rooms. Those with higher similarity scores are more likely to be paired together.
* Students have the ability to view each type of room. Admins can edit, add, and remove room images.
* Students can add people to their group indicating that they want to dorm together.
* Students with accessibility accommodations have priority room selection ensuring that they receive the room they need. Admins can override a student's room preferences.
* Students can cancel their room application.
* Should no rooms be available, students can enter a waitlist for their desired room.
* Admins have the ability to view how many people are in a dorm, on a waitlist, and applied.

2.3 Branching Policies

We use GitHub to share and edit files for this project. GitHub includes a main branch for integration and release. Each member has their own branch to be worked on, before being integrated into the main branch.

2.4 Project Management and Control

The Suite Dreams team will have at least one meeting each weekend on Discord and one meeting the day before a deadline. This will monitor our progress during each task. The meeting that takes place before a deadline will allow for final edits in the respective project.

## 3. Development Requirements

3.1 Hardware Requirements

* Since Suite Dreams is a web-based application, development can be carried out on any modern operating system, including MacOS, Windows or Linux. These platforms are fully compatible with the tools required for this project.

3.2 Software Requirements

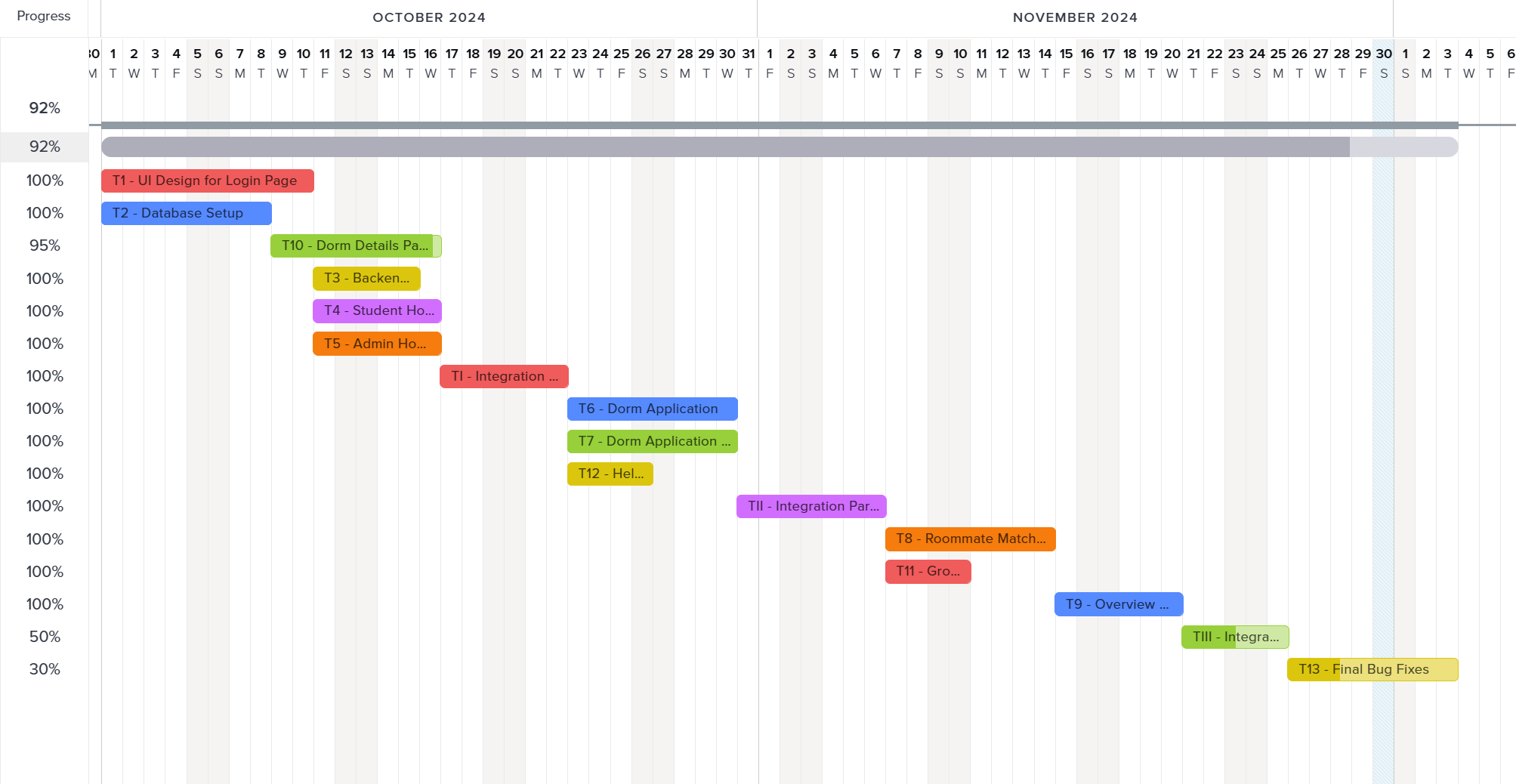
* Development will be done using the PyCharm IDE, where each developer will set up local environments with the required libraries and frameworks, including Python, React, Django and Firebase. Github will be used for managing version control throughout the project, and Firebase will be used for managing the database for the application.

## 4. Task Overview

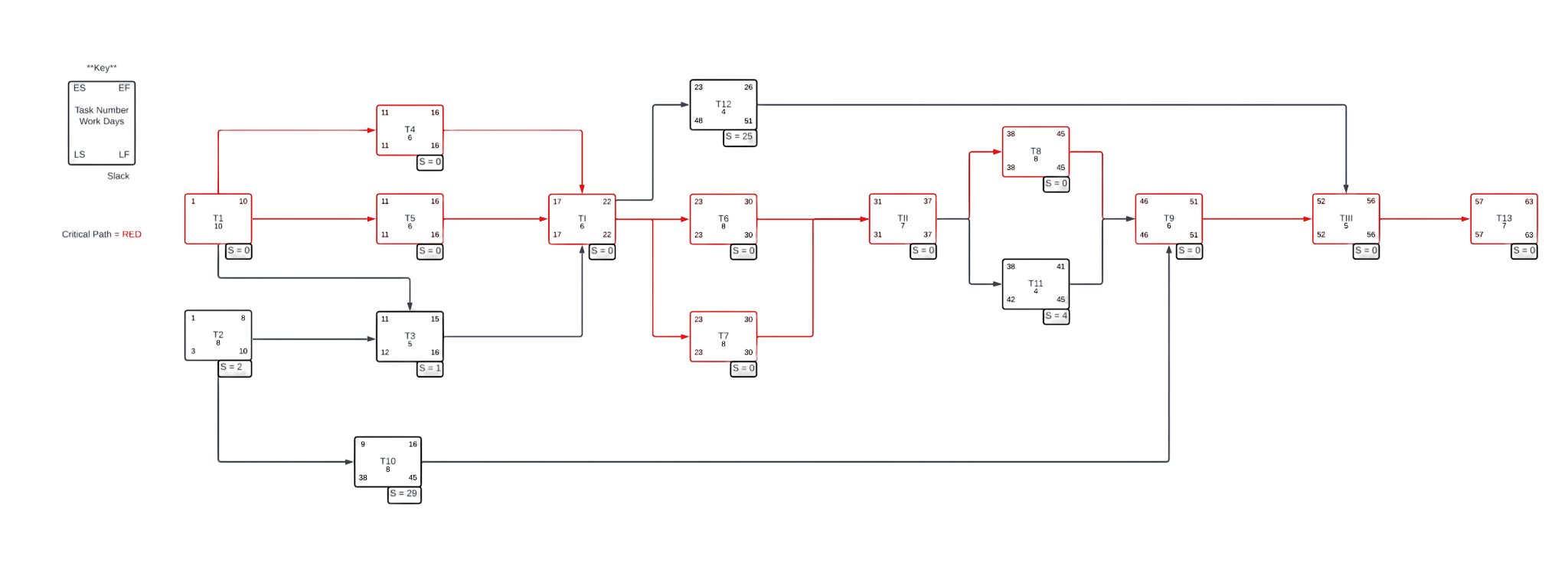
4.1 Task Table

| **Task ID** | **Task Description** | **Assigned To** | **Assistants** | **Frontend/Backend** | **Dependencies** | **Estimated Days** | **Start Date** | **End Date** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | UI Design for Login Page | Jack, Jonathan, Kyren, Arman |  | Frontend | - |  | Oct 1 | Oct 10 |
| T2 | Database Setup | Parsa, Alexes |  | Backend | - |  | Oct 1 | Oct 8 |
| T10 | Dorm Details Page | Jonathan, Kyren, Alexes | Parsa | Frontend/Backend | T2 | 8 | Oct 9 | Oct 16 |
| T3 | Backend Authentication Setup | Parsa, Arman | Kyren, Alexes | Backend | T1,T2 | 5 | Oct 11 | Oct 15 |
| T4 | Student Homepage | Jack, Jonathan | Parsa, Kyren, Alexes | Frontend | T1 | 6 | Oct 11 | Oct 16 |
| T5 | Admin Homepage | Jack, Jonathan | Parsa, Kyren, Alexes | Frontend/Backend | T1 | 6 | Oct 11 | Oct 16 |
| TI | Integration Part I | All Team Members | N/A | Frontend/Backend | T3,T4,T5 | 6 | Oct 17 | Oct 22 |
| T6 | Dorm Application | Jack, Kyren, Alexes, Jonathan | Parsa | Frontend | TI | 8 | Oct 23 | Oct 30 |
| T7 | Dorm Application Backend | Parsa, Kyren, Alexes, Jonathan, Arman | Jack | Backend | TI | 8 | Oct 23 | Oct 30 |
| T12 | Help Pages | Kyren, Alexes | Jonathan | Frontend/Backend | TI | 4 | Oct 23 | Oct 26 |
| TII | Integration Part II | All Team Members | N/A | Frontend/Backend | T7, T6 | 7 | Oct 31 | Nov 6 |
| T8 | Roommate Matching Algorithm | Alexes, Kyren, Parsa | Jonathan | Backend | TII | 8 | Nov 7 | Nov 14 |
| T11 | Group/Roommate Add Feature | Kyren, Jonathan | Alexes, Jack, Parsa | Frontend/Backend | TII | 4 | Nov 7 | Nov 10 |
| T9 | Overview Page (Admin & Student) | Jonathan, Parsa, Jack | Alexes, Kyren | Frontend/Backend | T8, T10,T11 | 6 | Nov 15 | Nov 20 |
| TIII | Integration Testing | All Team Members | N/A | Frontend/Backend | T9,T12 | 5 | Nov 21 | Nov 25 |
| T13 | Final Bug Fixes | All Team Members | N/A | Frontend/Backend | TIII | 7 | Nov 26 | Dec 3 |

## 5. Project Schedule

5.1 Gantt Chart

5.2 PERT Chart



## 6. Risk

As we initiate the Suite Dreams project, it is crucial to recognize potential risks that could affect the overall success of the endeavor. Although we cannot foresee every conceivable risk, we will concentrate on the most likely challenges that our team may encounter. By tackling these risks proactively, we can facilitate more efficient development and deployment stages.

6.1 Technical Risks

61.1 Database Performance and Scalability

Risk: With the increase in user numbers, the database may find it difficult to manage the demand, resulting in sluggish response times or even system crashes, particularly during high-traffic periods such as the commencement of a semester or during dorm application deadlines.

Mitigation:

* Employ database optimization strategies like indexing, caching, and refining queries.
* Explore options for database sharding or partitioning to alleviate the load.
* Continuously monitor and adjust database performance through tools such as Firebase performance monitoring.
* Activate auto-scaling for databases and servers to accommodate unforeseen traffic surges.

6.1.2 Integration Challenges

Risk: Challenges in integrating different components (frontend, backend, database, and third-party services) may result in delays, operational issues, or discrepancies in data.

Mitigation:

* Execute comprehensive integration testing at every phase of development to guarantee seamless interaction between components.
* Utilize well-documented APIs and adhere to established best practices for API design.
* Ensure consistent and clear communication among frontend, backend, and database teams to swiftly address integration challenges.
* Incorporate automated testing for API endpoints to detect issues at an early stage.

6.1.3 Security Vulnerabilities

Risk: Possible security breaches could jeopardize sensitive student information, including personal details and dorm preferences, potentially leading to legal consequences and harming the university's reputation.

Mitigation:

* Adopt strong security protocols, which encompass encryption (both at rest and in transit), secure authentication methods (e.g., OAuth, JWT), and routine security assessments.
* Establish role-based access control (RBAC) to ensure that only authorized personnel can access sensitive information.

6.1.4 Matching Algorithm Performance

Risk: The dorm and roommate matching algorithm may not operate efficiently, particularly with a large student population, causing delays in room assignments or erroneous matches.

Mitigation:

* Enhance the matching algorithm for efficiency by utilizing effective data structures and algorithms.
* Evaluate the algorithm using substantial datasets to confirm its scalability.
* Implement logging and monitoring to detect and rectify any performance issues in real-time.

6.2 Project Management Risks

6.2.1 Timeline Pressure

Risk: Urgent deadlines may result in hastily executed development, which can compromise quality, leave features unfinished, or lead to insufficient testing.

Mitigation:

* Develop a practical project timeline that includes buffer periods to handle unexpected delays.
* Employ Agile methodologies (such as Scrum) to divide the project into smaller, manageable sprints, facilitating ongoing progress monitoring and adjustments.
* Consistently monitor progress with project management tools like Trello, and modify resources or deadlines as necessary.

6.2.2 Resource Constraints

Risk: A shortage of skilled developers, designers, or hardware resources may impede development, particularly if essential team members are unavailable or if the necessary infrastructure is lacking.

Mitigation:

* Perform comprehensive resource planning at the project's outset to pinpoint potential bottlenecks.
* Evaluate the option of outsourcing certain tasks (like UI design or database management) if internal resources are insufficient.
* Keep a flexible development timeline that can be modified according to resource availability.
* Cross-train team members to ensure that several individuals are capable of managing critical tasks.

6.3 Compliance Risks

6.3.1 Data Privacy Regulations

Risk: Failure to comply with data privacy regulations could result in legal issues, fines, and reputational damage.

Mitigation:

* Regularly consult with legal experts to ensure compliance with all relevant data privacy regulations.
* Implement data protection measures such as encryption, anonymization, and secure data storage.
* Ensure that users can easily access, modify, or delete their data in accordance with privacy laws.

6.3.2 Accessibility Standards

Risk: The system may not meet required accessibility standards for students with disabilities.

Mitigation:

* Incorporate accessibility features (e.g., screen reader support, keyboard navigation, color contrast options) from the design phase.
* Follow WCAG (Web Content Accessibility Guidelines) to ensure the system is accessible to all students.

6.4 External Risks

6.4.1 Changes in University Policies

Risk: Sudden changes in university housing policies or procedures could require significant modifications to the system, leading to delays or additional costs.

Mitigation:

* Design the system with flexibility in mind, allowing for easy updates to policies and procedures.
* Document all policy-related system logic to ensure that changes can be implemented quickly and efficiently.

6.4.2 Third-party Dependencies

Risk: Issues with third-party services (e.g., Firebase, Bootstrap) or libraries could impact system functionality, availability, or performance.

Mitigation:

* Carefully evaluate and select reliable third-party components with strong community support and regular updates.
* Maintain updated documentation on all third-party dependencies, including their versions and known issues.
* Have contingency plans in place for critical dependencies, such as alternative services or libraries that can be used in case of failure.

6.5 Additional Risks

6.5.1 Data Loss or Corruption

Risk: Data loss or corruption could occur due to hardware or software failures, human error, or natural disasters, leading to loss of critical information such as student applications or room assignments.

Mitigation:

* Implement regular automated backups of all critical data, and store these backups in multiple locations (e.g., cloud storage and local storage).
* Use data redundancy techniques such as RAID for hardware storage systems to protect against hardware failures.

6.5.2 System Downtime

Risk: Unexpected outages or downtime could disrupt critical processes such as dorm applications or room assignments, especially during peak usage periods.

Mitigation:

* Implement redundancy measures such as load balancing and failover systems to minimize downtime.
* Schedule regular maintenance during off-peak hours to minimize the impact on users.

By identifying and addressing these potential risks early on, we can proactively minimize their impact and ensure the successful development and deployment of the Suite Dreams project.

## 7. Quality Assurance

7.1 Testing

Testing is vital in the development of Suite Dreams. Some testing used involves:

* Integration Testing: Each new section of code that is added to the existing main branch will be extensively tested, in order to guarantee functionality between the previous code and the new code.
* Regression Testing: Performed once a new section of code is integrated into the system. It is used to make sure that any new changes or additions to the code have not affected previously working sections.
* Compatibility Testing: Suite Dreams is intended to work on both Windows and Mac. It is used to guarantee compatibility on both systems.
* Performance Testing: Due to the use of a complex dorm allocation system and algorithm, efficient performance is vital and is tested regularly to ensure that it is at an optimal efficiency and speed.
* User Acceptance Testing: Testing involving users to make sure the application is accessible and intuitive. It is also used to make sure that the program works as intended.