Person in Context Assessment

**Final Report**

WSU Psychology Clinic



**Mentor:**

Dr. Scott & Belinda Lin

A silhouette of a person's head with a brain

AI-generated content may be incorrect.

Shaylin Smith, Christian Manangan, Jiaming Chu

CptS 423 Software Design Project II

Spring 2025

**TABLE OF CONTENTS**

[I. Introduction 4](#_Toc194871402)

[II. Team Members & Bios 5](#_Toc194871403)

[III. Project Requirements Specification 5](#_Toc194871404)

[III.1. Project Stakeholders 6](#_Toc194871405)

[III.2. Use Cases 7](#_Toc194871406)

[III.3. Functional Requirements 10](#_Toc194871407)

[III.3.1. Pica Assessment 10](#_Toc194871408)

[III.3.2. Phone Application 11](#_Toc194871409)

[III.4. Non-Functional Requirements 12](#_Toc194871410)

[IV. Software Design 13](#_Toc194871411)

[IV.1. Architecture Design 13](#_Toc194871412)

[IV.1.1. Overview 14](#_Toc194871413)

[IV.1.2. Subsystem Decomposition 15](#_Toc194871414)

[IV.2. Data Design 22](#_Toc194871415)

[IV.3. User Interface Design 23](#_Toc194871416)

[V. Test Case Specifications and Results 25](#_Toc194871417)

[V.1. Test Overview 25](#_Toc194871418)

[V.2. Environment Requirements 27](#_Toc194871419)

[V.3. Test Results 28](#_Toc194871420)

[V.3.1. Preliminary Tests 28](#_Toc194871425)

[VI. Projects and Tools Used 29](#_Toc194871426)

[VII. Description of Final Prototype 29](#_Toc194871427)

[VII.1. PDF Report Generator Subsystem 30](#_Toc194871432)

[VII.1.1. Functions and Interfaces Implemented 30](#_Toc194871437)

[VII.2. Summary of Prototype Demonstration 33](#_Toc194871438)

[VII.3. Client Comments and Suggestions 33](#_Toc194871439)

[VII.4. Design Modifications Based on Testing and Feedback 33](#_Toc194871440)

[VIII. Product Delivery Status 34](#_Toc194871441)

[IX. Conclusions and Future Work 34](#_Toc194871442)

[XI.1. Limitations and Recommendations 34](#_Toc194871447)

[XI.2. Future Work 35](#_Toc194871448)

[X. Acknowledgements 35](#_Toc194871449)

[XI. Glossary 35](#_Toc194871450)

[XII. References 36](#_Toc194871451)

[XIII. Appendix A – Team Information 36](#_Toc194871452)

[XIV. Appendix B – Testing Strategy Reporting 36](#_Toc194871453)

[XV. Appendix C – Project Management 36](#_Toc194871454)

[XVI. Appendix D – Additional Use Cases 37](#_Toc194871455)

# Introduction

Mental health has become one of the most pressing issues worldwide, with the World Health Organization listing mental health issues as the leading cause of global disease burden. Whether it is depression, anxiety or other mood disorders, these issues have a profound impact on individuals' daily lives, work productivity and interpersonal relationships. As people's awareness of the importance of mental health continues to increase, more and more technological solutions are being used to help individuals better manage and monitor their emotional health. According to the Global Mental Health Market Study, the demand for digital mental health solutions is expected to grow rapidly in the next few years, driving further development of mental health interventions around the world [1].

The combination of technology and mental health interventions provides individuals and clinicians with unprecedented opportunities to obtain real-time, personalized feedback and use data-driven approaches to improve treatment outcomes. However, there is still a lack of research on how personality affects treatment response. Although some studies have shown that patients' personality or temperament may affect their response to treatment, more specific evidence is needed to support this claim. Therefore, our team is committed to further verifying this hypothesis through this project - taking patients' personality traits into account can help develop more targeted treatment plans, thereby improving treatment effectiveness.

We aim to enhance the user experience by further developing and integrating two existing applications. The first is a web survey based on Qualtrics that generates personalized psychological assessment feedback for users. The second is a mobile application that allows users to self-monitor emotional events. Our goal is to optimize these apps so that they not only provide clear and timely feedback but also improve the contextual recommendation capabilities of self-monitoring apps through more sophisticated algorithms. Ultimately, our team aims to integrate these two systems into a more comprehensive mental health support platform that helps users better manage their emotions and behaviors while providing clinicians with deeper insights.

This project is being developed in collaboration with our mentor, Dr. Walt Scott, a Professor and Director of Psychology Clinic, and Belinda Lin, a graduate student in the Department of Psychology. Dr. Scott’s lab focuses on measuring personality, emotion, and goals through tests based on current personality science [2]. Their goal with this project is to create a system that enhances mental health treatment and clinical decision-making based on a patient's personality and emotional patterns.

The WSU Psychology Clinic is a university-based nonprofit organization staffed by advanced student clinicians. They serve children, adolescents, adults, and veterans in the Palouse region and provide psychotherapy and psychological assessment services. Their address and phone number are listed below

* Psychology Clinic, Washington State University, Johnson Tower 362, PO Box 6, Pullman, WA 99164-4820
* Phone: 509-335-3587

# Team Members & Bios

Shaylin Smith is a Senior at Washington State University and will graduate with a degree in Computer Science in Spring 2025. Shaylin has an interest in data science and analytics as well as maintaining databases. She has experience with Python, C++, HTML, and CSS as well as strong leadership and communication skills. She plans on using this skill set to learn more and grow to succeed in the future. For this project she will be responsible for maintaining the database for the project as well as working with the Qualtrics API and PICA PDF layout.

Christian Manangan is a computer science student interested in data science and software testing. Christian is currently a senior who will graduate in the Spring 2025 with a B.S. in Computer Science with a Math minor. Christian’s skills include C++, C#, Python, R, and Java. His prior projects included building and analyzing sequence similarity networks (SSNs) to study relationships in cancer gene sequences and creating a Java-based financial application with an emphasis on testing frameworks like Mockito, Spring MVC, and Selenium. For this project, his responsibilities will include generating personalized PDF reports based on survey responses for the client’s lab team to review and provide feedback, as well as integrating working files into the PythonAnywhere web application.

Jiaming Chu is a computer science student who is interested in developing games and AI. Her skills include C++, Java, and Python. Her previous projects include independently making a small game using Unity. For this project, her responsibilities included integrating and creating a Flask application: merging two separate files (Qualtrics and generator\_report) together. The goal was to automatically trigger the function after processing the survey data. And modifying the initial PDF layout: adding and moving different chapters and texts based on the client's expectations and requirements so that the latest PDF layout better meets the client's requirements

# Project Requirements Specification

There are many different types of psychological treatment for those with mental health issues and it can be difficult to figure out what treatment best suits each person. One way to approach finding the best treatment for a patient is by factoring in the person’s personality and looking into how their temperament affects their thoughts, emotions, and behavior in different contexts. The WSU Psychology Clinic is trying to utilize this personality-based approach and investigate a patient’s temperament and responses to everyday situations and try to build the best treatment based upon those results. Through this project, the WSU Psychology Clinic can adopt a more evidence-based approach and show how a person’s personality effects how well they respond to different types of treatment.

This is where the Person in Context Assessment (PICA) comes into play. The focus of this project is split among two separate Qualtrics surveys used to collect more information about a patient’s temperament. The first survey is one that asks questions about if the user has any goals in different areas of their life such as work, relationships, and personal health as well how much the user is working towards that goal and how happy they are about it. Then the results of the user’s temperament are calculated, and a PDF of the results are sent to the email. Our first course of action will be to fix the previous Python code that emails the PDF results to the user. Right now, the user must go into PythonAnywhere to be able to fetch their results, but we want to make sure that their results can be emailed to them. Also, we will be looking into editing the format of the PDF and changing the graphs and results to make sure that it is understandable to the user and looks visually appealing.

The second survey that is used in PICA is one that is used by the patient daily. The user will input their experiences that day and then will also fill in their thoughts, feelings, goals, and actions for the event. Then, emotionally similar events are grouped together and then the clinician can look at both the patient’s temperament as well as how they reacted to these events and use that personality-based data to give a more personalized mental health treatment plan. With this second survey, we want to work on integrating the first survey so that the user would be able to see their original PICA results inside of this daily survey. The clinician should be able to enter the patient’s PICA results into the daily survey and have the user be able to see the PDF of their results in the daily survey app. Also, the daily survey should be able to tailor questions to the user based upon their temperament results and better predict how they may react to daily situations.

Finally, with the daily survey we will be looking into creating a more complex clustering algorithm. We need to be able to group a user’s experiences based upon how similarly the patient thought or felt during the event and that is done based upon what feelings the patient fills in on a checkbox list. Any events that have over 50% shared feelings will then be categorized as the same. We want to either move towards using more complex math to group the different experiences more accurately or possibly investigate utilizing the ChatGPT API to cluster the data. The user can fill in the checkboxes for how they were feeling but they can also write any specific thoughts or emotions into a textbox. The current simpler clustering algorithm does not factor this text box in when clustering so the patient or clinician would have to go in themselves to the survey and read the text box and manually cluster the events if they see the need to. By utilizing AI and ChatGPT, we would remove the need to manually read and cluster those events that have information stored in the text box and instead the AI could read the text boxes and suggest which events should be clustered.

## Project Stakeholders

Our primary customer is the Washington State University (WSU) Psychology Clinic, where the project will be used to help clinicians better understand their patients' personality traits and provide personalized treatment recommendations. Our mentors, Dr. Scott and Belinda Lin, are the primary contacts for the project and will provide guidance on how to optimize the personality assessment feedback and self-monitoring applications. The project will ultimately be used by clinical graduate students and clients at the WSU Psychology Clinic as a tool to assess the relationship between patient personality and treatment response. In addition to WSU, we have established potential partnerships with the Portland Psychology Clinic, where clinicians are also interested in using this system to help their patients.

Potential customers include the Portland Psychology Clinic and other psychotherapy institutions. To attract the interest of these institutions, our software project needs to ensure easy integration and use processes. The interface of the personality assessment report and self-monitoring application must be clear and easy to use, report generation and data processing must be efficient, and the feedback content can directly support clinical decision-making. To further promote application, the project should have cross-platform compatibility so that these institutions can easily deploy the system into their existing technical infrastructure.

Finally, all stakeholders in the project will benefit from a system with a clear structure, concise code, and easy extensibility. This will help clinicians better integrate these tools into their daily work and provide a solid foundation for future system improvements. Our development team will prioritize the needs of Dr. Scott's team, but throughout the design and development process, we will also pay attention to how to meet the needs of other potential customers and institutions to ensure the broad applicability and sustainability of the project.

This document aims to be a summary of our project’s current progress and technical details of how we have constructed the project with examples of our use cases, testing methods, and project architecture.

## Use Cases

The use cases describe common scenarios of user interactions with the system, explaining how various functional requirements are applied in specific situations. Figure 1 below, is a use case diagram that illustrates the core interactions between different system users and the system.

A diagram of a person's work flow

Description automatically generated

Figure 1: Use case diagram

The descriptions of a couple of use cases are given below. See Appendix D for the rest of the use cases descriptions.

**Use Case 1: Complete Survey**

|  |  |
| --- | --- |
| Use Case | Complete Survey |
| Actors | Participant |
| Pre-condition | Participant logged in and started the Qualtrics survey |
| Post-condition | The participant completed all survey questions and the system received the complete survey results. |
| Main Flow | - Participant opens the Qualtrics survey.  - Participant answers all questions and submits the survey.  - The system confirms the survey is complete and ready to generate feedback. |
| Alternative Flow | -If a participant drops out of the survey midway, the system saves the current progress and allows the participant to continue later. |
| Related Requirements | FR1:Results PDF  FR2:Automatic PDF Sending  FR3:Data Collection |

**Use Case 2: Generate Personalized Feedback**

|  |  |
| --- | --- |
| Use Case | Generate Personalized Feedback |
| Actors | System |
| Pre-condition | The survey was completed and the system received the participant's survey results. |
| Post-condition | Personalized assessment feedback was generated, including preliminary feedback and detailed feedback. |
| Main Flow | -The system receives and analyzes survey data from participants.  -The system generates preliminary feedback, including personality assessment results.  -The system generates detailed feedback, providing specific treatment strategies. |
| Alternative Flow | -If the data is incomplete, the system will generate feedback containing only basic information. |
| Related Requirements | FR1:Results PDF  FR5:PICA Based Questions  FR6:Clustering Algorithm  FR7:Data Collection |

**Use Case 3: Generate and Send PDF Report**

|  |  |
| --- | --- |
| Use Case | Generate and Send PDF Report |
| Actors | System |
| Pre-condition | Personalized feedback generated. |
| Post-condition | PDF reports were generated and emailed to participants and clinicians. |
| Main Flow | -The system generates a personalized PDF report based on the feedback, including charts and treatment recommendations.  -The system automatically sends the PDF report to the email address provided by the participant.  -The system confirms that the report was successfully sent. |
| Alternative Flow | -If the email fails to be sent, the system will record the failure information and try again. |
| Related Requirements | FR1:Results PDF  FR4:PICA Assessment Results  FR7:Data Collection |

## Functional Requirements

Functional requirements are specific features that a software system must have to perform its intended purpose. Functional requirements defines what the system should do by outlining its behavior, actions, and outputs based on end user needs. Each functional requirement is listed below with a detailed description, source, and priority level.

### Pica Assessment

**Results PDF:**

|  |  |
| --- | --- |
| Description | The PICA Qualtrics survey needs to be able to create a PDF of the resulting charts and graphs that is easy to understand for the user as well as adding more in-depth details and a radar chart. |
| Source | Required from Client. Adding on to already implemented functionality. |
| Priority | Priority Level 0: Essential and required functionality |

**Automatic PDF Sending:**

|  |  |
| --- | --- |
| Description | The PICA Qualtrics survey results will be automatically sent to the email that the user provided in the survey and will be sent after the survey is completed. |
| Source | Required from Client. Client needs to be able to send results to patient. |
| Priority | Priority Level 0: Essential and required functionality |

**Data Collection:**

|  |  |
| --- | --- |
| Description | The PICA survey data from all surveys taken must be sent to the MongoDB database to be stored. |
| Source | Internal requirement from team. Building upon functionality added by previous team. |
| Priority | Priority Level 0: Essential and required functionality |

### Phone Application

**PICA Assessment Results Integration:**

|  |  |
| --- | --- |
| Description | The PICA data for the patient must be able to be inputted into the phone app which allows the user to see their results in the phone app. |
| Source | Requirement from Client. Client needs to be able to integrate the PICA data with the phone app. |
| Priority | Priority Level 0: Essential and required functionality. |

**PICA Based Questions:**

|  |  |
| --- | --- |
| Description | The phone app will ask specific questions based upon the patient’s temperament that was entered into the survey. |
| Source | Requirement from Client. Client needs to be able to ask questions based upon the entered temperament. |
| Priority | Priority Level 0: Essential and required functionality. |

**Clustering Algorithm:**

|  |  |
| --- | --- |
| Description | The algorithm that is used to group events that are “functionally equivalent” based on shared thoughts, feelings, or behaviors will be grouped together either through a mathematically complex algorithm or AI and will be then approved to be clustered by the user or the doctor. |
| Source | Internal requirement from team. Building upon already implemented functionality. |
| Priority | Priority Level 2: Extra features or stretch goals |

## Non-Functional Requirements

Non-functional requirements outline the operational qualities of the system, such as performance, system availability, maintainability, and security, and require accuracy and security of data stored in the MongoDB database. In addition, the system needs to ensure user experience to ensure that it meets quality standards beyond core functionality. Details of non-functional requirements are as follows.

|  |  |  |
| --- | --- | --- |
| **Non-Functional Requirements** |  | **Description** |
| [NFR-1]System Performance |  | The system should be able to scale as the number of users grows, especially when a large number of clinical graduate students or psychological clinics use the system at the same time. |
| [NFR-2]Data Storage and Management |  | All emotional events, evaluation data, and user feedback should be accurately stored in a MongoDB database, and the integrity of the data should be ensured. The database should support automatic backup and recovery mechanisms to ensure that no data is lost in the event of a system failure. |
| [NFR-3]Security |  | The system should have role-based permission management capabilities to ensure that only authorized users (such as clinicians and researchers) can access specific assessment results and user emotional events. |
| [NFR-4]System Availability |  | If the system fails, it should have automatic recovery and error handling mechanisms to ensure that the user experience is not affected. For example, when PDF generation fails, the system should automatically retry or send a notification to the administrator. |
| [NFR-5]User Experience |  | The application should have an intuitive user interface so that users can start using the system without having to read a lot of documentation. The emotional event recording and feedback functions in the mobile self-monitoring application should be simple and easy to understand, and participants can complete them quickly. |
| [NFR-6]Maintainability |  | The system should be designed with an extensible and maintainable architecture to support the addition or modification of future functions. The development team should be able to quickly locate and fix potential problems in the system and update it without affecting users. |

# Software Design

The purpose of this section of the document is to outline the architecture, a bird’s eye view of the system, as well as components and system functionality for the project. This section is intended for the WSU Psychology Clinic, project developers, and stakeholders involved. It will serve as a guide for the continuing development and integration of the existing survey and self-monitoring application.

## Architecture Design

The following sections go into detail on the system’s architecture and key subsystems. The architecture design describes how different components interact with one another and the subsystem decomposition breaks down each functional unit.

### Overview

The current project builds upon the architecture and development efforts of previous iterations, focusing on integrating and improving two primary applications: 1) a Qualtrics-based web survey report that generates personalized personality assessments, and 2) a mobile self-monitoring app that logs emotional events daily. Our architectural design follows the client-server model, which ensures scalability, modularity, and reliable interaction between the client-side user interfaces and the server-side logic that handles data processing, storage, and analysis.

The client-server model was chosen for several reasons. First, this pattern allows for clear separation of concerns: the client focuses on user interaction and inputs, while the server handles data processing, sentiment analysis, and storage. Given the large volume of data and the need for event-based functionality (logging, monitoring, and real-time feedback), this model ensures efficient data handling and user experience. It also facilitates the integration of advanced features, such as treatment recommendations and clustering algorithms in the mobile app, making it flexible for future enhancements.

The system consists of several key components that work together to provide real-time, personalized feedback to users. On the client side, users interact with the Qualtrics-based survey and self-monitoring mobile app to input their emotional events or complete personality assessments. These inputs are processed by the server, which stores the data securely, analyzes it using sentiment analysis and other algorithms, and generates relevant feedback. The feedback is then delivered back to the client as personalized reports or recommendations.

The UML components, shown in Figure 2, illustrate the interaction between client-side (UI) and server-side components (application logic, DBMS, and security), as well as the flow of data in the system. The diagram highlights how different modules interact to achieve the goals of the system, from user input to data processing and feedback generation.

**A diagram of a system

Description automatically generated**

Figure 2: Architecture Design

## Subsystem Decomposition

Here we describe the different components of our system and how they interact with each other.

**[UI Handler]**

**Description**

The UI Handler is responsible for managing all interactions between the user and the system. It handles input from both the web-based Qualtrics survey and the mobile self-monitoring app. This subsystem ensures that users can submit their survey responses and emotional event logs, and it relays this information to the appropriate back-end components for further processing. It also manages the output by displaying personalized feedback or reports generated by the system.

**Concepts and Algorithms Generated**

The UI Handler does not utilize complex algorithms but instead focuses on input validation and user interaction. It uses basic validation mechanisms to ensure that the input data (survey responses, logs) is in the correct format and passes the necessary checks before being sent to the system. The selection of this solution was driven by the need for an efficient and responsive user interface that can process user inputs in real-time and provide an intuitive experience.

**Interface Description**

**Services Provided:**

|  |  |  |
| --- | --- | --- |
| Service Name | Service Provided To | Description |
| Input Collection | Input Handler | Collects and validates user input (survey data or emotional events) and forwards it to Input Handler. |
| Display Feedback | Application Logic | Displays feedback or reports generated by the system back to the user. |

**Services Required:**

|  |  |
| --- | --- |
| Service Name | Service Provided From |
| Processed Feedback | Application Logic |
| Input Validation | Input Handler |

**[Environmental Handler]**

**Description**

The Environmental Handler manages external data interactions, ensuring that the system can adapt to the environment it operates in. It gathers data from sources such as mobile sensors, location data, and other external APIs, which is used to enhance the user experience in the self-monitoring app. For example, it might adjust recommendations based on the user's environment.

**Concepts and Algorithms Generated**

The Environmental Handler utilizes location-based algorithms to detect user context and adjust app behavior. It uses a combination of external APIs and real-time data gathering techniques. The decision to use these algorithms was based on the need to provide a dynamic and context-aware experience for users, especially in the self-monitoring app.

**Interface Description**

**Services Provided:**

|  |  |  |
| --- | --- | --- |
| Service Name | Service Provided To | Description |
| Location Data | Application Logic | Provides location-based information to adjust app behavior or user feedback. |
| External Data Updates | Input Handler | Supplies data from external APIs, like weather or location data, to augment user inputs. |

**Services Required:**

|  |  |
| --- | --- |
| Service Name | Service Provided From |
| Input Processing | Input Handler |
| External API Data | External Sources |

**[Input Handler]**

**Description**

The Input Handler processes all user inputs submitted through the UI. It ensures the correct format and validity of the data before sending it to the Application Logic for further processing, such as report generation or emotional analysis.

**Concepts and Algorithms Generated**

The Input Handler employs data validation algorithms to ensure that all inputs conform to the expected format and values. This was chosen to maintain the integrity of data entering the system and prevent errors during further processing.

**Interface Description**

**Services Provided**:

|  |  |  |
| --- | --- | --- |
| Service Name | Service Provided To | Description |
| Validated Input | Application Logic | Sends validated user inputs (survey or emotional logs) for further processing. |
| Data Preprocessing | Sentiment Analysis Algorithm | Prepares user input data for sentiment analysis. |

**Services Required**:

|  |  |
| --- | --- |
| Service Name | Service Provided From |
| User Input Collection | UI Handler |
| Environmental Data | Environmental Handler |

**[IDE Reader]**

**Description**

The IDE Reader collects data from the Integrated Development Environment (IDE) or external file sources, and supplies it to the relevant system components, such as for analysis or report generation.

**Concepts and Algorithms Generated**

No specialized algorithms are used here beyond standard file reading and parsing techniques. The primary objective is to accurately retrieve and transfer data.

**Interface Description**

**Services Provided**:

|  |  |  |
| --- | --- | --- |
| Service Name | Service Provided To | Description |
| Data Input | Application Logic | Reads data from files or external sources and forwards it for processing. |

**Services Required**:

|  |  |
| --- | --- |
| Service Name | Service Provided From |
| Data Storage Access | Database Management |

**[Database Management System]**

**Description**

The Database Management System (DBMS) stores all user data, including survey responses and logs from the self-monitoring app. It ensures that data is efficiently stored, retrieved, and updated as needed, while providing security measures to protect user privacy.

**Concepts and Algorithms Generated**

The DBMS uses standard database indexing and encryption techniques to handle large volumes of data securely and efficiently. The trade-off considered was between performance and security, resulting in the selection of algorithms that prioritize data integrity and protection.

**Interface Description**

**Services Provided**:

|  |  |  |
| --- | --- | --- |
| Service Name | Service Provided To | Description |
| Data Storage | Application Logic | Provides secure storage for user inputs and logs, as well as survey results. |
| Data Retrieval | Sentiment Analysis Algorithm | Retrieves relevant user data for analysis. |

**Services Required**:

|  |  |
| --- | --- |
| Service Name | Service Provided From |
| Processed Data Storage | Application Logic |
| Data Access | IDE Reader |

**[Sentiment Analysis Algorithm]**

**Description**

This subsystem performs sentiment analysis on the data collected from users. It evaluates survey responses and daily logs, extracting emotional patterns and providing insights that are used in generating feedback and recommendations.

**Concepts and Algorithms Generated**

The sentiment analysis algorithm relies on natural language processing (NLP) techniques to extract emotional cues from user inputs. This allows for personalized feedback based on users' emotional states. The algorithm chosen was based on its accuracy and ability to handle large-scale input data efficiently.

**Interface Description**

**Services Provided**:

|  |  |  |
| --- | --- | --- |
| Service Name | Service Provided To | Description |
| Sentiment Analysis | Application Logic | Provides emotional analysis based on user inputs, assisting in generating personalized reports. |

**Services Required**:

|  |  |
| --- | --- |
| Service Name | Service Provided From |
| Input Data | Input Handler |

**[Security Component]**

**Description**

The Security Component handles the security of user data throughout the system, providing encryption, authentication, and secure communication between subsystems to ensure that all user interactions and data storage remain protected.

**Concepts and Algorithms Generated**

The Security Component uses encryption algorithms (such as AES) and authentication protocols (OAuth) to ensure secure access and data transmission. Trade-offs between performance and security were considered, with a focus on safeguarding sensitive user data without significantly impacting system performance.

**Interface Description**

**Services Provided**:

|  |  |  |
| --- | --- | --- |
| Service Name | Service Provided To | Description |
| Data Encryption | Database Management | Ensures all stored data is encrypted and secure from unauthorized access. |
| User Authentication | UI Handler | Manages login credentials and ensures that only authorized users can access the system. |

**Services Required**:

|  |  |
| --- | --- |
| Service Name | Service Provided From |
| User Data Encryption | Database Management |

## Data Design

Currently we are storing our databases in MongoDB Atlas that was established by the previous team. This allows the client to be able to have access to the database and easily add our team and any future teams to the project. This MongoDB database holds all our data related to the web app that collects the user’s daily logs of their thoughts and feelings. In this database we store users, surveys, signatures, situations, and all multiple-choice options as collections in the database. Each survey document contains an ID, user, signature, fi-then, situation, positive feelings, negative feelings, positive thoughts, negative thoughts, and behavior. The web app then makes a connection to the cluster of our database and then is ready to use. The positive and negative thoughts and feelings as well as behavior options each are stored in the database and then appear as multiple-choice options in the app. The web app then retrieves these multiple-choice options from the database and displays them on the screen when a person is logging into an event. The document classes for these multiple-choice options are stored in Models.py with each collection in Mongo having a class in the main app file psychclinic.py which fills the collections when the app is initialized. Once the app is run, the multiple-choice options are filled out and do not need to be filled out again. If we need to update the multiple-choice options, we can easily do so through MongoDB by adding them into the database. All of this can be visualized using an Entity Relationship (ER) Diagram seen in Figure 3.

A diagram of a diagram

AI-generated content may be incorrect.

Figure 3: ER Diagram

The diagram represents the database schema with entities, attributes, and relationships.

Another way we store our data is through a csv file we create when retrieving the survey data from Qualtrics, which hosts the PICA. When we call Report\_Generator.py after someone completes their PICA, the report generator then connects to the Qualtrics API to collect the list of surveys taken which is then locally saved as a csv file in PythonAnywhere. That csv file is then loaded into our report generator as a DataFrame. We take the last row of the DataFrame, which is the data of the most recently taken survey and use that data to generate our report and send it to the corresponding email. Storing the data this way allows us to have local access to the list of surveys and potentially send multiple survey reports to multiple people based off the one csv file.

## User Interface Design

The user interface was originally designed and developed by a capstone team prior to the previous one and we have adhered to the same color schemes and icons they created. The previous capstone team made updates to the home page which includes icons for each page in place of text. They also changed the dashboard to only display home and logout.

The login screen, Figure 4, is still the same as it was when the capstone team prior to the previous one created it.

**A screenshot of a login form

Description automatically generated**

Figure 4: Login

Users have two options when creating an account: an admin account or a regular user account. After entering the required information and creating credentials, they can log in. The previous team discussed this login page with the client, and it was concluded that no changes were necessary. The user’s name is displayed after logging in which is based on information from their login, as shown in Figure 5.

**A screenshot of a computer

Description automatically generated**

Figure 5: Home Page

The team prior to the previous updated the landing page which allows users to start a new journal entry. The user can then begin a journal entry by selecting either a positive or negative entry. This takes them through additional pages with relevant questions and prompts which is shown in Figure 6 below.

A screenshot of a computer

Description automatically generated

Figure 6: Logging an Event

Reviewing past journal entries is another important function shown in Figure 7.

A screenshot of a computer

Description automatically generated

Figure 7: Past Situations

From the admin’s perspective, admins have access to see every user’s survey. Admins have the option to view surveys and utilize the Qsort tool after signing in (Figure 8).

A close-up of a blue and white screen

Description automatically generated

Figure 8: User Surveys

The “view survey” option lets them view surveys for all users (Figure 9).

A screenshot of a computer

Description automatically generated

Figure 9: User Qsort Entry

The other option is Qsort, which allows admins to assign or insert a survey to specific users.

Many use cases will utilize these interfaces for user interaction. Use case 4: Log Emotional Event, the journal entry pages are vital for users to log their emotional events. Use case 5: View Functionally Equivalent Situations, the past journal entries page will be used when users want to view similar emotional situations that were determined by the system. Use case 6: Import Personality Assessment Results, the admin login as well as the survey views will be used by clinicians to important and manage personality assessment results.

# Test Case Specifications and Results

A structured and thorough testing process has been established for the Person in Context Assessment system to ensure reliability and accuracy. The following sections outline the specification of the tests, environment requirements, and test results.

## Test Overview

To ensure that the Person in Context Assessment system functions reliable, we structured our testing process to cover various testing types such as unit testing, system testing, and integration.

Our testing approach is designed to ensure each component in the Qualtrics based survey and the self-monitoring mobile app functions efficiently, consistently, and accurately. We will confirm that each feature of the system aligns with the appropriate specifications and operates reliably. This will be done using a methodical and iterative process and guide us through every step of the testing lifecycle.

* **Identify the Requirements to be Tested**: Using the Software Requirements Specification, we will create each test case to make sure it satisfies both functional and non-functional requirements. This way, we will be able to prioritize the essential features like PDF generation, automated email delivery, and clustering algorithms.
* **Determine Test Cases for Each Module**: We will create particular test cases for each module to verify its functionality. These test cases will be linked to the relevant requirements.
* **Review Test Data and Test Cases**: Before test cases and data are executed, we will review them to make sure they adequately cover the anticipated operating scenarios and edge cases. This will help in identifying any gaps or inconsistencies early in the testing process.
* **Define Expected Results**: For every testing input, it must have a well-defined expected output. For example, the email the survey participant inputs must be the email the PDF will be sent to.
* **Document Test Configuration and Data**: How the test was set up, test data, and expected outputs will all be documented in a test plan. This will allow future potential teams to reproduce and replicate it as well as use it as a reference.
* **Performing the Tests**: Following the test plan, each test case will be executed.
* **Document Tests from Performance**: For tests that failed, we will write a report that outlines the problem, potential causes, and the series of events leading up to the failure.
* **Perform Integration and System Testing**: Integration testing will allow us to see if the system components or wiring interact properly. This is to ensure the entire application functions as a unified entity.
* **Continuous Integration (CI)**: Since we are running this project in an iterative nature, we will use CI. This will allow us to automate testing for new code changes. We will also be able to maintain code quality and identify issues early.
* **Test Documentation and Reports**: A detailed report will be prepared upon completion of the testing phases. This will include test results, setups, and any problems discovered.

The team will use the Python testing framework for unit testing. Each class and method will be unit tested to ensure that it works properly. We will first unit test each new feature and then move on to integration testing. For this project, unit testing may include but is not limited to:

* Test that the modules that generate reports can correctly process inputs and generate expected outputs
* Test that the chart generator can display data correctly
* Verify that the data collection and PDF generation modules work together properly
* Check that the automatic email sending function can be triggered and send reports correctly

We will perform these tests manually or automatically using continuous integration tools. We will prioritize integration testing for the most important features and then expand to other features. For this project, integration testing may include but is not limited to:

* Confirm that the application can correctly handle and respond to user input
* Test whether the report generation module can be seamlessly integrated with other functional modules
* Check whether the system can provide a consistent user experience in different environments

To ensure that programs and applications run smoothly, we need to test the entire code as a whole, looking for various possible interruptions and problems. System testing may evolve as new components are added, but overall system testing will be an iterative process to ensure that the application functions smoothly. System testing will include functional testing, performance testing, and user acceptance testing to ensure that user needs are met under various requirements.

Within the scope of functional testing, we will check the overall progress and usage of the application. For this application, validating that participants complete the survey and successfully generating and sending the report are important functional testing aspects. In addition, checking the accuracy of charts and data in the generated PDF file, the validity of user input, etc. are also key testing elements.

To ensure optimal performance and optimization of the application, we will test aspects such as data storage methods, efficiency, and data structures. Performance testing will process large amounts of user data to ensure that data is stored in an optimal manner and retrieved efficiently when requested by users. In addition, we will conduct stress testing to verify that the system remains stable under high load.

User acceptance testing will ensure that the application is ready for user usage. This testing will verify whether the application interface is easy to use, backend tasks are performed without errors, user requirements are met, and the system can be deployed and run successfully.

## Environment Requirements

For hardware requirements, we need a computer that is capable of running our program. This computer must have sufficient processing power and memory to run our web application as well as a stable network to be able to access MongoDB, PythonAnywhere, and Qualtrics.

In terms of system software, we need an operating system that supports Python since that is the programing language we have used for our project. Also, we need an environment that can support MongoDB for our database.

Our testing environment will be relying on specific application development tools. We will be using Pytest as well as the Github CI pipeline to host our testing. The Github framework will provide a testing pipeline that allows our team and future developers to ensure the tests run and pass before committing and merging the work.

## Test Results

\*We have not done any formal testing yet. We plan to focus on testing and deployment for sprint 7. We will document the test cases with the format below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aspect being tested** | **Expected Result** | **Observed Result** | **Test Result** | **Test Case Requirements** |
| What are you testing and how are testing it? | What is the expected result? (Before you run the test) | What is the observed result? (After you run the test) | Pass or Fail? | What specific settings are required to replicate this test case? You do not have to list the default project setup values here |



### Preliminary Tests

**Unit Tests**

Report Generation

* Successfully tested the generation of PDF reports with sample survey data.
* Verified that graphs and descriptive text appear correctly in the output, adhering to the latest feedback from the client.

Graph Representation

* Tested bar graph labels and adjusted axes descriptors to ensure they match the intended format (e.g., “BAS-Drive” instead of “BAS-D”).

**Integration Tests**

Email Delivery

* Verified successful delivery of generated PDF reports to both participant and clinician email addresses using test accounts.

Qualtrics Data Workflow

* Tested the end-to-end process of receiving survey data via a webhook, generating the PDF report, and delivering it via email.

Findings and Improvements

* Identified the need to increase x-axis font sizes and reposition graph descriptions for better visibility.
* Detected inconsistent indentation and paragraph spacing, which are being resolved in subsequent iterations.

# Projects and Tools Used

|  |  |
| --- | --- |
| Tool/library/framework |  |
| Python | Python is used as the primary backend language to handle data processing, analysis, and server-side operations. |
| MongoDB | MongoDB serves as our database system to store and manage user responses, assessment data, and survey results. |
| Qualtrics | Qualtrics is utilized as the survey platform for delivering personality assessments and collecting response data. |
| Flask | Flask is a web framework for Python that allows users to build web applications. Flask hosts both the frontend and backend of our application. |
| PythonAnywhere | PythonAnywhere is the online hosting service for our report generator and phone app that allows users to access the phone app from their own mobile devices. |

|  |  |  |  |
| --- | --- | --- | --- |
| Languages Used in Project | | | |
| HTML | CSS | Python |  |

# Description of Final Prototype

The Alpha Prototype for our project focuses on developing and integrating critical components of the proposed architecture for the Psych Clinic Web Application, designed to enhance therapeutic interventions by collecting, analyzing, and presenting survey and journaling data from participants. The current prototype includes several key subsystems: the Qualtrics Survey Data Integration, which retrieves survey responses through a webhook and processes them for further use; the PDF Report Generator, responsible for creating personalized reports that summarize participants' personality assessment results with graphical representations, detailed textual feedback, and recommended treatment strategies tailored to individual needs; and the Email Delivery Subsystem, which automates the process of sending these reports to participants and clinicians upon survey completion. The User Interface (UI) has also been enhanced, with functional pages for login, survey submission, and report viewing, designed to improve usability for therapists and participants. Substantial progress has been made in integrating these subsystems, allowing for a seamless workflow from survey submission to report generation and delivery. Testing has been conducted at both the unit and integration levels. The PDF Report Generator has undergone iterative improvements based on client feedback, with approximately 80% of its functionality implemented, including adjustments to graph designs and formatting issues such as paragraph indentation and consistent font sizes. Integration testing confirmed successful end-to-end workflows, while UI tests gathered usability insights, leading to planned adjustments like increasing font sizes and refining button placements. Findings from these tests highlighted the need to address issues such as large text gaps in reports, unclear graph labels, and basic email template designs, all of which are currently being refined.



## PDF Report Generator Subsystem

Here we will outline how the final product of our PDF report generator will look and how it will perform.



### Functions and Interfaces Implemented

**Survey Data Integration**

The PDF Report Generator subsystem retrieves survey data submitted by participants through the Qualtrics survey platform. This data is then parsed and processed for report generation.

**Personalized Report Generation**

The subsystem generates personalized PDF reports based on survey responses, summarizing key personality assessment results. Reports include:

* Graphical representations of survey results (e.g., bar graphs).
* Recommended treatment strategies tailored to participants’ specific needs.
* Clear textual descriptions to ensure the report is user-friendly and informative.

Figures 9 and 10 illustrate the personalized PDF report before and after edits were made respectively.

A screenshot of a graph

Description automatically generated

Figure 9: PDF Report Goals Page Before Edits

A graph with a bar and a graph with a bar and a graph with a bar and a graph with a bar and a graph with a bar and a bar and a graph with a bar and

Description automatically generated with medium confidence

Figure 10: PDF Report Goals Page After Edits

**Email Delivery Functionality**

The generated PDF reports are automatically sent to both participants and clinicians via email upon survey completion.

**Improved Report Design**

Multiple iterations of the PDF layout and graphical content have been implemented to address client feedback.

Improvements include:

* Reformatting headers, paragraphs, and graph labels.
* Adjusting the design of graphs to enhance readability.

**Interface with Web Application**

The PDF Report Generator subsystem is integrated with the PythonAnywhere-hosted Flask web application. The generate\_report API endpoint was created to manually or automatically trigger report generation as needed.

**Remaining Work**

Refining the text formatting to eliminate large gaps between words and ensure consistency in font sizes. Further improving graph clarity, including label adjustments and resizing for better presentation.

## Summary of Prototype Demonstration

During our prototype demonstration, we showed multiple iterations of the PDF report that survey participants would receive upon completion of the survey. These reports were created for the lab team of our client to examine and offer thorough feedback. The demonstration’s main goal was to point out improvements in the layout, graph designs, and information formatting. This iterative process allowed us to repeatedly refine the PDF report based on feedback from the client and their lab team. This process also allowed us to ensure the report aligned with the client’s specific requirements and expectations for functionality and presentation.

## Client Comments and Suggestions

The lab team and client provided valuable feedback and suggestions. The points below provide a high-level overview of their feedback:

* Add page numbers and position them at the top after the title page
* Change the large gaps between words in the text to improve readability
* Indent the first lines of all paragraphs to more clearly demarcate paragraphs
* Center headers and ensure font style is consistent throughout the document
* Replace the “High Average” and “Low Average” y-axis descriptions on graphs with a single “Average” marker at 2.5
* Update the labels “BAS-D” to “BAS-Drive”, “BAS-FS” to “BAS-Fun Seeking, and “BAS-R” to “BAS-Reward” to improve clarity
* Adjust the width of bar graphs to span most of the page as well as center them
* Move the graph descriptions below the graphs except for the “Personal Goals and Standards” graphs, where the current side text placement remains appropriate
* To differentiate between zero scores and missing scores, add a marker labeled “Missing” on the x-axis before “0”
* Increase the font size of the x-axis descriptors for better visibility
* Use complete spellings of the ranking of values (EducLearning to Education/Learning)

## Design Modifications Based on Testing and Feedback

Based on the client and lab team feedback, we plan to implement the following design changes:

* Reformatting of paragraphs and headers
* Adjusting graph size, labels, and descriptions
* Add a “Missing” marker on the x-axis
* Ensure consistent font size for key elements
* Refine the report generation function/process to address the spacing issues in the text

These changes are crucial in aligning with the client’s expectations of the prototype while also providing a clearer presentation and story of the data for survey participants. Further testing cycles and sprints will validate these changes, and additional feedback will be incorporated as necessary.

# Product Delivery Status

For our project, the PICA report generator will be handed off within our second sprint during the semester and the phone app will be handed off at the end of the spring semester. We have demoed the report generator for each iteration when we make progress to ensure that the client is happy with our progress and once we are fully finished, we will add our completed files to the PythonAnywhere project where the PICA report generator and phone app are hosted.

As we have worked on our product, we have hosted our files on the repo at https://github.com/plasmaaShark/ACME16-PSY-FullStackApp/tree/main to host our source control. The files in MongoPsychClinicWeb host our phone app and can be locally ran by running “pip install -r requirements.txt” in the terminal and then running “python psychclinic.py” in the terminal. The terminal will then show a link that the user can click to open the web app on their browser. The files in PsychClinic-ReportGenerator host the report generator and can also be run locally. To run the report generator, the user just has to input “python Report\_Generator\_Sorting.py” which will then send the report for the most recent survey taken to the email at the end of the create\_report() function.

# Conclusions and Future Work

As the Person in Context Assessment system evolves, it is important to discuss its current limitations as well as areas for improvement.



## Limitations and Recommendations

While the Person in Context Assessment system effectively combines personal assessments and self-monitoring tools, it has some limitations.

* **Basic Emotional Event Grouping**. The system groups emotional events based on the number of checkbox selections selected. If two emotional events share 50% of the checkbox selections. It does not process free-text responses to do this grouping. Using Natural Language Processing model to process free-text responses is a more sophisticated way of grouping emotional events.
* **Performance & Scalability Issues**. After a user completes a survey, the system may take 5 – 10 minutes to deliver and email the personalized pdf report back to the user. As more users interact with the system, some may not even receive their report. The system currently retrieves the latest survey result to produce a report. The system also allows us to pick a range of survey results. For example, we can create reports for the latest five survey results. A recommended approach is having a way to select specific survey results based on certain constraints.
* **Limited Customization for Reports**. The system generates a static personalized pdf report. This adds limitations if a clinician wants to edit or annotate reports before sending them to their patients. A recommended approach is to have a feature where clinicians can add notes, adjust interpretations, and change data visualizations like the graphs

## Future Work

The Person in Context Assessment has successfully developed and combined important features to improve psychological assessments and self-monitoring tools for clinical use. Using a Qualtrics-based survey system and mobile self-monitoring app, we created a system that aids clinicians better understand patients’ emotions and personalities

The current system successfully automates sending an in-depth personalized report based on survey results and allows patients to track emotional patterns overtime. However, clinical psychology and technology are always improving, so there are many ways to improve and extend this project. Future work should include enhancing the clustering algorithm for grouping functionally equivalent emotional events and ensuring compliance and meeting legal standards for data protection.

The current system groups emotional events based on keyword matching or the number of same options describing the emotional event. Using an AI-driven algorithm or model that categorizes events based on emotional complexity would be a better route.

Finally, ensuring compliance and meeting legal standards for data protection is the last future work to consider. For HIPPA compliance, the system needs to implement data encryption and role-based access controls. We also need the system to have automated logs that track data access and modifications.

# Acknowledgements

We sincerely appreciate all those who contributed to the success of this project. We want to thank our mentor, Dr. Scott, for his guidance, patience, and valuable feedback in developing this system. We are also grateful for Belinda Lin for her help in debugging the early issues of the system.

Additionally, we would like to thank the WSU Psychology Clinic for providing this opportunity and to make a system that is impactful and directly supports those who are in the mental health field. We also acknowledge our sponsors for providing the necessary resources and support, which allowed us to develop and test our system effectively. Thank you for making this project possible.

# Glossary

**ChatGPT:** Stands for Chat Generative Pre-Trained Transformer. It is a large language model-based chatbot that allows users to refine and steer a conversation towards desired length, format, style, etc.

**Qualtrics:** Qualtrics is a cloud-based platform that allows users to create, distribute, and analyze surveys. It is the platform used to host the two surveys of the project.

**Python:** Python is a high-level, general-purpose programming language that is known for its flexibility and vast number of free libraries.

**PythonAnywhere:** PythonAnywhere is an online integrated development environment (IDE) and web hosting service that is targeted for the Python programming language.

**MongoDB:** MongoDB is an open-source, NoSQL database management program that can be accessed remotely.

**Temperament:** In psychology, temperament refers to an individual’s innate behavioral and emotional style that influences how they react to the world.

**HTML:** HTML stands for HyperText Markup Language and is the standard language for creating webpages and defining their structure and content.

# References

|  |  |
| --- | --- |
| [1] | TBRC Business Research PVT LTD, “Digital Mental Health Market Shows Rapid Growth, Projected To Grow At A Rate Of More Than 16% As Per The Business Research Company's Digital Mental Health Global Market Report 2024,” *Globe News Wire,* Mar. 25, 2024 [Online]. Available: <https://www.globenewswire.com/news-release/2024/03/25/2851807/0/en/Digital-Mental-Health-Market-Shows-Rapid-Growth-Projected-To-Grow-At-A-Rate-Of-More-Than-16-As-Per-The-Business-Research-Company-s-Digital-Mental-Health-Global-Market-Report-2024.html> [Accessed: Apr. 8, 2025]. |
| [2] | W. D. Scott, S. Paup and C. Kirchhoff, "Clinical Application of Social Cognitive Theory: A Novel Personality Assessment Procedure and a Case Study of Personality Coherence," *European Journal of Personality,* vol. 36, pp. 371-390, 2022. |
| [3] | D. C. Rettew and L. McKee, “Temperament and Its Role in Developmental Psychopathology,” *Harv Rev Psychiatry,* vol. 13, no. 1, pp. 14-17, Jan-Feb 2005. [Online]. Available: PubMed Central, <https://pmc.ncbi.nlm.nih.gov>. [Accessed Apr. 8, 2025]. |

# Appendix A – Team Information

|  |  |  |
| --- | --- | --- |
| A person smiling for a picture  Description automatically generated | A person taking a selfie  Description automatically generated | A person taking a selfie  Description automatically generated |
| Shaylin Smith | Christian Manangan | Jiaming Chu |

# Appendix B – Testing Strategy Reporting

\*We will list the majority of testing outputs here so that the “Test Results” section won’t be cluttered with too many of them

# Appendix C – Project Management

Our team’s weekly schedule typically focuses on meeting with our clients every week on Tuesday so we can have a clear objective on what to work on for the next week. We do these meetings on Zoom and keep notes for each meeting so we can reference what was talked about and what goals we should focus on. After having our meeting with the client, we will meet as a team over Discord and discuss how we want to divide the work for the week. This is very beneficial as it sets expectations on what each team member will be doing and helps us assign work items on our Kanban board to each member. Some of the planning documents we have used are the GitHub projects and issues. We have set up a Kanban board to display our issues.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 11: Kanban Board

# Appendix D – Additional Use Cases

**Use Case 4: Log Emotional Event**

|  |  |
| --- | --- |
| Use Case | Log Emotional Event |
| Actors | Participant |
| Pre-condition | Participants were logged into a mobile phone self-monitoring application. |
| Post-condition | Emotional events and related information have been saved to the database |
| Main Flow | -The participant opens the mobile app and chooses to record an emotional event.  -The participant enters the type of emotion, thoughts, feelings, and behavior information.  -The system saves the record and confirms that it was saved successfully. |
| Alternative Flow | -If the device is offline, the system will automatically sync the data when the connection is restored. |
| Related Requirements | FR5:PICA Based Questions  FR6:Clustering Algorithm FR7:Data Collection |

**Use Case 5: View Functionally Equivalent Situations**

|  |  |
| --- | --- |
| Use Case | View Functionally Equivalent Situations |
| Actors | Participant, System |
| Pre-condition | View Functionally Equivalent Situations |
| Post-condition | The system provides participants with functionally equivalent situational suggestions based on the recorded data. |
| Main Flow | -The system analyzes the emotional events recorded by the participants.  -The system generates functionally equivalent situational suggestions based on a clustering algorithm.  -The system displays the suggestions to the participants. |
| Alternative Flow | -If the system fails to identify similar situations, participants will be prompted to record more events to obtain more precise suggestions. |
| Related Requirements | FR5:PICA Based Questions  FR6:Clustering Algorithm FR7:Data Collection |

**Use Case 6: Import Personality Assessment Results**

|  |  |
| --- | --- |
| Use Case | Import Personality Assessment Results |
| Actors | Clinician |
| Pre-condition | A PDF report of the personality assessment results has been generated and sent. |
| Post-condition | The clinician has entered the assessment results into the mobile app. |
| Main Flow | -Clinician logs into mobile app.  -Clinician manually enters or uploads personality assessment results.  -System confirms data was saved successfully. |
| Alternative Flow | -If the data is incorrect, the system will prompt the doctor to re-enter or modify the data. |
| Related Requirements | FR4:PICA Assessment Results FR6:Clustering Algorithm FR7:Data Collection |

**Use Case 7: Collect User Feedback**

|  |  |
| --- | --- |
| Use Case | Collect User Feedback |
| Actors | Researcher |
| Pre-condition | Participants and clinicians completed the corresponding functional operations. |
| Post-condition | User experience data is stored in the database for subsequent analysis. |
| Main Flow | -Researchers regularly push experience surveys to users.  -Users complete the experience surveys and submit data.  -Researchers review the data for subsequent analysis and improvement. |
| Alternative Flow | -If the user skips the survey, they will be prompted to fill it out again the next time they use the system. |
| Related Requirements | FR6:Clustering Algorithm FR7:Data Collection |