# **Cover Page**

PlannerBird

Kamile Vaicekonis

CSC 3150—Systems Design—Professor Andy Cameron

**System Specification—5/25/24**

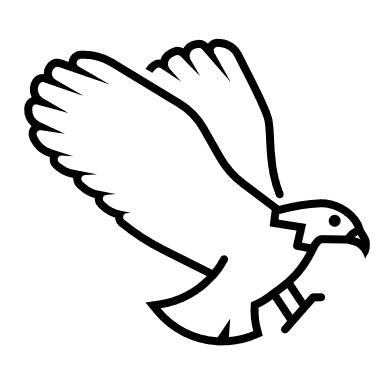


Table of Contents

[Cover Page 1](#_Toc168324166)

[1. Executive Summary 3](#_Toc168324167)

[2. Introduction 4](#_Toc168324168)

[2.1. Problem Statement / Project Vision 4](#_Toc168324169)

[2.2. System Capabilities 4](#_Toc168324170)

[2.3. Non-functional Requirements and Design Constraints 5](#_Toc168324171)

[2.4. System Evolution 5](#_Toc168324172)

[2.5. Document Outline 7](#_Toc168324173)

[3. Structural Model 8](#_Toc168324174)

[3.1. Model Introduction 8](#_Toc168324175)

[3.2. Class Diagrams 8](#_Toc168324176)

[3.3. Metadata 9](#_Toc168324177)

[4. Architecture Design 22](#_Toc168324178)

[4.1. Architecture Overview 22](#_Toc168324179)

[4.2. Infrastructure Model 22](#_Toc168324180)

[**4.2.1.** **Deployment Diagram 1 – Architecture Overview** 22](#_Toc168324181)

[**4.2.2.** **Deployment Diagram 2 – Nodes and Artifacts** 23](#_Toc168324182)

[4.3. Hardware and Software Requirements 23](#_Toc168324183)

[**4.3.1.** **Hardware Components** 24](#_Toc168324184)

[**4.3.2.** **Required Software Components** 25](#_Toc168324185)

[4.4. Security Plan 25](#_Toc168324200)

[**4.4.1.** **Security Overview** 25](#_Toc168324201)

[**4.4.2.** **Security Plan** 26](#_Toc168324202)

[5. User-Interface 27](#_Toc168324203)

[5.1. User-Interface Requirements and Constraints 27](#_Toc168324204)

[5.2. Window/Screen Navigation Diagram 27](#_Toc168324205)

[5.3. UI Wireframes 29](#_Toc168324206)

[5.4. Reports: "Formal Output" Design 39](#_Toc168324207)

[6. Appendices 40](#_Toc168324208)

[6.1. Glossary 40](#_Toc168324209)

[6.2. References / Bibliography 41](#_Toc168324210)

[6.3. Supporting documentation 41](#_Toc168324211)

# **Executive Summary**

My name is Kamile Vaicekonis, and I am a student at Seattle Pacific University, I am working on the software application PlannerBird. To summarize, PlannerBird aims to make travel planning easy by guiding its users through creating a travel plan for their next trip. The application will have a search functionality that recommends tailored destinations based on user preferences. They will also be able to create detailed travel plans with schedules. There will be a budgeting feature that allows the user to specify their intended expense from the trip and compare it to the estimated cost that is calculated within the application based on their plan. The client of this application is any individual that likes to travel. This application is inclusive of those who struggle to travel because of planning, people that have difficulty coming up with places to visit, or people that just want to go explore and try something new. PlannerBird is an application for anyone that likes to go on adventures.

As this document is created for the developers, it is important to highlight that this application is solely designed to be a cross-platform mobile application. Because people travel with their phones, it makes the most sense for it to be a mobile application that is easily accessible to its users at any point in the day. Accessibility is especially important for productivity software because the users will be more inclined to use the software if it is easy to

At this point in time, PlannerBird has not been implemented yet. Currently, the design process for the applicated has been completed and this document has been created as an outcome of that. Hopefully after reviewing this material, feedback will fine tune the final details related to the completion of this project’s design stage.

# **Introduction**

PlannerBird is a mobile application that allows users to create and store their personal trip plans. This software will allow the user to create an itinerary for their travels while also recommending places to visit. The user will take a preferences survey that will allow the software’s search algorithm to suggest relevant locations to visit along their journey. The application has 11 primary functionalities that will be outlined within this section, introducing the project. There will be references made to the System Proposal document, which goes into greater detail for each functionality. This section will be summarizing what was explained within the proposal.

## **Problem Statement / Project Vision**

When traveling, it is difficult to keep track of places to visit, stay within a budget, and find the best route to reach your destination. Most of the time, travel information is scattered  
across several notes applications, hidden in emails or text messages, and saved in various  
opened tabs in your browser. The information spreading across multiple platforms can cause  
many problems, such as missed flights, forgotten landmarks, unexpected costs, and a  
dissatisfactory trip outcome. Traveling is widely considered one of the most significant  
expenses for the average adult. The opportunity that I am proposing is to merge all travel plans into one mobile application called PlannerBird, that is accessible to its users at any point during vacation. PlannerBird will reduce the scatteredness of trip planning and will not require physical planners or papers, which often frustratingly take up space in our packed baggage.  
 Another common issue when traveling is ignorance of events, destinations, and restaurants that tourists enjoy. Finding landmarks to visit takes time and effort. Not all people enjoy researching places to visit or learning historical information about their destination. Some travelers are families that want to find attractions for their kids. Others might unexpectedly find themselves in a new city with a desire to find things to do in the area. There are also people wanting to spend more time exploring their hometown. People are multifaceted, with unique interests and circumstances that should be tailored when traveling. Therefore, I propose PlannerBird will also personalize destination search results based on surveys taken by the user. The goal is for PlannerBird to be customized for a family taking a road trip, a businessperson on a work trip, or a couple on their resort vacation at the beach. In response to this vision, PlannerBird will provide its users with a custom, all-encompassing, mobile travel planning application, allowing users to create and store itineraries, budgets, and routes. PlannerBird will organize travel plans, all while suggesting personalized destinations.

In terms of scope, this project will include search algorithms, data personalization, database storage, and network connection. Creating a social media platform for users to share

photos from trips is outside the bounds of this application. The aim of this application is not

to create a social media platform; instead, it is to build productivity software that will help

users create an effective travel plan. Anything outside of the realm of productivity is not within

the scope of this application. Stakeholders of PlannerBird include travelers, investors, various businesses and establishments that will be affected by PlannerBird suggestions, competing applications, and the tourism business.

## **System Capabilities**

The following is a list of PlannerBird’s expected system capabilities summarized:

1. The search algorithm should display suggested destinations to visit, taking into account the user’s preferences. Functional Requirement 1
2. The database must store and update all user general preferences for destinations. Functional Requirement 1
3. The list of suggestions should be categorized by preference selection first and advertising second. Functional Requirement 1
4. The algorithm must search the internet for places to visit and automatically categorize

them. Functional Requirement 1

1. User login information must be private and well-protected. Functional Requirement 2
2. The user will remain signed in after account creation, device will be recognized and saved to reduce logins. Functional Requirement 2
3. The user may opt in or out of data sharing with advertisers. Functional Requirement 3
4. Users can create a modifiable list for their travel plans and store data and time-related information for their trip. Functional Requirement 6
5. Each travel plan will allow the user to specify a budget goal for the trip. Functional Requirement 7
6. The expense of the trip will be automatically calculated based on the events and restaurants saved on the trip plan. Functional Requirement 7
7. Each travel plan must be archivable, the archives must be accessible by the user at any time if they wish to unarchive a plan. Functional Requirement 8

## **Non-functional Requirements and Design Constraints**

PlannerBird has various constraints that affect the system. The biggest constraint being security. Most of PlannerBird’s value comes from the database that stores user preferences. The search algorithm for recommending destinations to visit is what makes the application unique compared to similar applications already on the market. There are many other constraints that affect the feasibility of PlannerBird. There are no issues with feasibility that should be of significance to the development of the system.

## **System Evolution**

The future evolution of PlannerBird in terms of system versions will be explained in the following section.

* + 1. **Version 2 Changes**

In version 2 of PlannerBird, the goal is to increase functionality in the application. PlannerBird’s aim is to be productivity software for travel planning. This means that the application must have well-implemented features that aren’t frustrating to use. One of the most important functionalities is improving the search algorithm and application personalization.

1. Personal trip rating system
   1. The user can rate the trips that they have completed
      1. Ratings must be modifiable at any time
   2. The user can give feedback on suggested destinations that the search algorithm found based on preferences
      1. Database must adjust search algorithm based on feedback from the user about the suggestions
2. Packing list
   1. The user can create a packing list for each travel plan
      1. Each item to pack will appear in a list with checkboxes for each item, the user is able to delete things off this list and add new information
      2. Works similarly to a notes application
3. Document storage
   1. Sensitive user information is stored as files within the application
   2. Must be password protected by a user-set pin number
   3. Pin can be changed through double authentication
   4. Pin stays the same for every document storage section in each trip plan
4. Flight/car rental searchability
   1. Flights and car rentals can be searched after specifying a departing location, destination, and price range
   2. When a flight or rental is selected, the user is redirected to their mobile device’s browser to book the flight or rental
      1. **Version 3 and beyond Changes**

In version 3, PlannerBird will focus on usability of the application. This involves making improvements to the user interface and overall aesthetics of the application. After receiving customer feedback, the team will modify the basic application to be visually appealing and memorable to its users.

1. Improve user interface
   1. A color theme will be picked for the application
   2. Font of the application will be consistent
   3. Icons will not be as word-heavy, buttons should feel intuitive to the user without much description
2. Destination descriptions
   1. The application will display descriptions for historical landmarks and their significance
   2. Descriptions will be found from the internet and sourced when displayed for the user, so that they may go to the internet website where the information was derived from if they would like to find out more
   3. Descriptions will be shown when the user is searching for destinations and wants to learn more about the place before committing to traveling to the location.
3. PC website implementation
   1. Turn PlannerBird into a website that users can log into rather than simply a mobile application

## **Document Outline**

For the remainder of this document, the elements of PlannerBird’s system will be discussed in greater detail for its developers.

1. Structural model: This section will describe the structure of the software application through class diagrams.
   1. Model introduction
   2. Class diagrams
   3. Metadata
2. Architecture design: This section will describe how the physical system architecture and software architecture will be designed. It will also discuss a security plan the architecture that will be implemented to protect the system.
   1. Infrastructure model
      1. Deployment diagrams 1 – architecture overview
      2. Deployment diagrams – nodes and artifacts
   2. Hardware and software requirements
      1. Hardware components
      2. Required software components
   3. Security plan
      1. Security overview
      2. Security plan
3. User interface: This will go over all requirements and constraints for the user interface. It will show how the user will be able to navigate through the application. It will also provide a rough depiction of wireframes.
   1. Requirements and constraints
   2. Window/screen navigation diagram
   3. UI wireframes
   4. Reports: formal output design
4. Appendices: The remaining sections are self-explanatory
   1. Glossary
   2. Bibliography
   3. Supporting documentation

# **Structural Model**

## **Model Introduction**

The following section will outline the class diagrams that are necessary for PlannerBird software. As this document is mainly targeted towards the developers of this application, the introduction will be brief. Class diagrams will describe all attributes, methods, and associations between classes. There will also be discussion about the metadata that is related to the class diagrams. After going through this section of the document, expectations for coding this project should be somewhat straightforward. Though it is possible for changes to occur during the development process if things are to be improved or modified.

## **Class Diagrams**

The following image shows the PlannerBird Class Diagram with all the classes belonging to the application that will have to be created for the software to work properly.

A diagram of a planner bird

Description automatically generated

Link to the diagram:

<https://drive.google.com/file/d/1kmdEgrngapWrzxzMOpE6pl8njxQtmW3e/view?usp=drive_link>

## **Metadata**

The following section will go into greater detail for each object within the class. There will be additional information provided to explain the uses of each object within the application.

Class Diagrams - Table of contents

1. User pg 10
2. Plan pg 13
3. Itinerary pg 15
4. Route pg 17
5. Destination pg 18
6. Search pg 19
7. Advertisers pg 21
8. User

A screenshot of a computer

Description automatically generated

Description: The user class stores all personal user information and travel plans that are created by the user.

Visibility: public

Is abstract: no

Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Read-only | Multiplicity |
| Id | Each user has a unique id so that the database has a unique key for each user to have access to their travel plans | True | 1 |
| Name | Each user has a name that will be referred to within the application for personalization | False | 1 |
| Email | An account must be created with a unique email address, there is no username for this application | False | 1 |
| Password | A user has a unique secure password which they use to log in with on different devices | False | 1 |
| Phone | A phone number is required upon account creation for two factor authentication | False | 1 |
| PinAuth | The user gets a pin number texted or emailed to them which they use to log in with two factor authentication | True | 0..\* |
| travelPlans | The user’s travel plans for different trips | False | 0..\* |
| archivedPlans | The user’s outdated plans which they have archived manually | False | 0..\* |
| Preferences | The user’s travel preferences | False | 1 |
| dataSharing | The user’s preference on whether or not they will allow data collection from advertisers | False | 1 |

Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Query | Polymorphic |
| VerifyLogIn | This function verifies whether the credentials provided by the user are correct and allows them to log in or denies entry | Yes | No |
| SignOut | This function signs the user out of the account | No | No |
| CreateAccount | This function passes the user input values to the database so that the new account is created | No | No |
| UpdateAccount | This function allows the user to modify their account information | No | No |
| DeleteAccount | This deletes the user’s account | No | No |
| TwoFactAuthentication | This function validates the authentication of the pin that the user received | No | No |
| DataSharing | This function sets the user’s preference on data sharing | No | No |
| CreatePlan | This function creates a new travel plan for a trip by the user | No | No |
| SelectPlans | This function selects user specified travel plans so that they can be modified or deleted | Yes | No |
| ModifyPlan | This function modifies a single plan at a time | No | No |
| DeletePlans | This function allows the plans to be deleted | No | No |

1. Plan

A screenshot of a computer

Description automatically generated

Description: The Plan class stores all class information for a travel plan.

Visibility: public

Is abstract: no

Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Read-only | Multiplicity |
| tripName | This is the trip name that the user will see on the homepage | False | 1 |
| userBudget | The user may specify a budget for their trip | False | 1 |
| calculatedBudget | The calculated budget is automatically calculated based on which destinations are stored in the plan | True | 1 |
| Itinerary | This is an object that stores all the dates, times, and descriptions of the plan | False | 1 |
| Active | If active is set to true, the homepage will display that the current travel plan day itinerary | True | 1 |
| searchResults | This is a list of the results of the search that is sorted | True | 1 |

Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Query | Polymorphic |
| SetName | This function sets the name of the trip | No | No |
| SetBudget | This function sets the budget that was specified by the user for the trip plan | No | No |
| CalculateBudget | This function will calculate the estimated expense of the trip | Yes | No |
| Activity | This function will be set to true if the trip has started | No | No |
| SearchDestination | This function will be called when a search is being made | Yes | No |

1. Itinerary

A white box with black text

Description automatically generated

Description: This class stores the itinerary data for each travel plan.

Visibility: public

Is abstract: no

Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Read-only | Multiplicity |
| departDate | This is the departure date for the trip | False | 1 |
| returnDate | This is the return date for the trip | False | 1 |
| Time | This is the date and time storage for the destinations stored in a list | False | 0..\* |
| Destinations | This is a list of all the destination information | False | 0..\* |
| Route | This member stores the travel route plan | False | 0..1 |

Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Query | Polymorphic |
| SetDates | This function sets the dates for the duration of the trip | No | No |
| ModifyDates | This function allows for modification of the dates for the travel | No | No |
| ModifyItinerary | This function modifies the itinerary, which modifies the dates and times of destination arrivals | No | No |
| SetRoute | This function sets the route for the trip | No | Yes |

1. Route

A close-up of a route

Description automatically generated

Description: This class stores route information for the trip. It is still in a very rough stage, for future versions of PlannerBird, this class needs to be made better.

Visibility: public

Is abstract: no

Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Read-only | Multiplicity |
| Duration | This is the duration of the full trip, how many hours it will take | True | 1 |
| travelRoute | This member stores a list of doubles with geographical coordinates of locations that will be visited | False | 0..1 |

Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Query | Polymorphic |
| SetRoute | This function sets the route of the travel plan | No | Yes |
| ModifyRoute | This function modifies the route of the travel plan duration and travelRoute | No | No |

1. Destination

A white background with black text

Description automatically generated

Description: This class stores all restaurant, landmark, and event destination members that are significant to this application.

Visibility: public

Is abstract: no

Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Read-only | Multiplicity |
| Price | This member stores the price of visiting the destination | True | 1 (ranged) |
| Location | This member stores the geographical coordinates of the destination | False | 1 |
| Name | This stores the name of the destination | False | 1 |
| externalLink | This is for storage of an external link to the destination website (example: event page, restaurant booking page) | False | 0..\* |

Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Query | Polymorphic |
| Display | This function is called to display the destinations on the screen | No | No |

1. Search

A screenshot of a search engine

Description automatically generated

Description: This class handles all the search related information within the application.

Visibility: private

Is abstract: no

Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Read-only | Multiplicity |
| Search | This is the search string that was sent out by the user | False | 1 |
| searchResult | This is the list of destination objects that are matched to the search string | True | 1 |

Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Query | Polymorphic |
| DisplayResult | This function displays the result of the search that the user made on the application search results page | No | No |
| Sort | This function sorts the destination results before displaying them on the screen so that they are ordered with respect to user preferences and advertisers | No | No |
| MatchedSearch | This function searches the database and internet for searches that match the search string that the user provided upon clicking search | Yes | No |

1. Advertisers

A close-up of a list

Description automatically generated

Description: This class stores all of the advertiser information for search results

Visibility: private

Is abstract: no

Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Read-only | Multiplicity |
| destinationAds | This is a list of the destinations that PlannerBird is paid to push on the results page for matching search queries | False | 1 |

Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Query | Polymorphic |
| UpdateList | This function updates the destinationAds list to have the current list of advertisers included | No | No |

# **Architecture Design**

## **Architecture Overview**

In this section of the system specification, the goal is for the developers to get a closer look at the general architecture of the software application. There will be deployment models that help to visualize how the entire thing is going to come together. There will also be a list of software and hardware requirements for the mobile application. This section will be significant because a lot of the hardware will be outsourced to reduce costs of running the system. There will also be security measures discussed to protect all the assets of PlannerBird.

## **Infrastructure Model**

* + 1. **Deployment Diagram 1 – Architecture Overview**

A diagram of a network diagram

Description automatically generated

Link to diagram:

<https://lucid.app/lucidchart/6b45d0be-9ae7-4ee5-b46c-7c8e174e560a/edit?viewport_loc=148%2C49%2C1671%2C1062%2C0_0&invitationId=inv_97641ce2-a28e-4044-a988-43aea5ef78ff>

* + 1. **Deployment Diagram 2 – Nodes and Artifacts**

A diagram of a computer server

Description automatically generated

Link to diagram:

<https://lucid.app/lucidchart/8ca37ada-cd46-48da-8a40-f826c9808410/edit?viewport_loc=-372%2C-380%2C2354%2C1496%2C0_0&invitationId=inv_34e8504b-b85a-4045-840d-2641003504d6>

## **Hardware and Software Requirements**

The following information will outline the hardware and software that is necessary for PlannerBird to function. PlannerBird requires specific hardware and software due to the way that the application requires a database to store user information and preferences. The server performs actions on the database within the server, not the mobile device that the user is using. There should be communication that is sending out requests to the server through the mobile application. There should always be response listeners to requests made from the application to hear back from the database and server system.

* + 1. **Hardware Components**

PlannerBird is a mobile application that will require certain versions or higher of Apple and Android. It will have a remote server to handle all database hardware. This is because the scope of the project may not be determined until the application is deployed and people begin using it. PlannerBird would like to outsource its servers for the project so that, if necessary, servers can increase as more users appear on the application. Because of this growth model, PlannerBird will be using AWS cloud computing for all of the servers for the application. mySQL will be used as the relational database to store user information.

1. AWS
   1. This is the main cloud server that PlannerBird will be using to process all requests made by the user
   2. The server will communicate with the database through queries
2. mySQL
   1. This is the relational database that will be storing all user information
   2. It will communicate with the server to create the perfect search suggestions for the user
      1. **Required Software Components**

The following list describes all software components that will be necessary for PlannerBird developers:

1. Clion
   1. This is the IDE that the project will be using to work on the project. This is due to C++ being object-oriented, which suits this project well
   2. Also, the developers working on this project are most proficient in C++.
2. Git
   1. This is a wonderful collaborative tool that allows developers to track all changes that were made to code along its process of development.
3. mySQL
   1. This is the database that will be used for the application that will store all user data, preferences, login information, and travel plans
   2. It will be able to perform queries on the database when searches are requested
4. Internet
   1. To run through a list of destinations in the area, the application must have connection to the internet so that it can accurately display updated information on destinations that are recommended
5. Cross-platform mobile application
   1. PlannerBird requires iOS 15 or Android 5.0 devices and up
   2. PlannerBird is NOT accessible through desktop computer until version 3 of the application is released

## **Security Plan**

* + 1. **Security Overview**

PlannerBird collects user data regarding their travel preferences, which is an asset that must be protected from being exploited. The other asset to PlannerBird is the algorithm that suggests places to visit based on the preferences. This section of the System Specification explains different threats to security of PlannerBird. I have listed the biggest threats to security of the application here:

* Unauthorized access to user data
* Physical theft
* Theft of login credentials
* User data leakage/breach
* Malware
* Virus
* Natural disasters: fire, flood, loss of power, circuit failure
  + 1. **Security Plan**

In response to the possible threats to security, I have created a list of controls that will be applied to each type of security threat if one were to occur. These are open for modification as the system develops and better practices are discovered to protect data. No matter what, the security of the application is of utmost importance when releasing it to the public.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Components | Natural Disasters |  |  |  | Malicious Attacks |  |  |  |
|  | Fire | Flood | Loss of power | Circuit failure | Virus | Malware | Data breach | Theft |
| Servers | 9 | 9 | 9 | 9 | 5, 6 | 5, 6 | 1, 2, 3, 5, 6 | 1, 6, 7, 8, 4 |
| Hardware | 9 | 9 | 9 | 9 | 5, 6 | 5, 6 | 1,5, 6, 7, 8, 4 | 1,5, 6, 7, 8, 4 |
| Database | 9 | 9 | 9 | 9 | 5, 6 | 5, 6 | 2, 3, 5, 6, 7, 10 | 1, 6, 7, 8, 4 |
| AWS cloud storage | 9 | 9 | 9 | 9 | 5, 6 | 5, 6 | 2, 3, 5, 6, 10 | 1, 6, 7, 8, 4 |

Controls:

1. Encourage user to reset their password every 12 months
2. Only allow complex passwords that include both upper-case and lower-case as well as symbols and numbers
3. Multi-factor authentication by asking a security question when the device is not recognized by the application
4. Flag suspicious log-ins from unrecognized devices
5. Encryption of data
6. Consistent penetration testing by InfoSec team and analysis
7. Allow users to opt in or out of data sharing with advertisers
8. Enforce authorized access to sensitive user data only by users and advertisers (if opted in)
9. Perform regular backups of the database
10. Firewall

# **User-Interface**

## **User-Interface Requirements and Constraints**

This section will go over all the important graphical user interface components of the system. For this aspect of the application, the guiding principle is to keep it simple and straightforward. It is undesirable to clutter up the visual space of the application with unneeded labels and titles for things that could be described with simple iconography. There should also be a homepage that is easy to navigate to at any point within the application navigation. The homepage should nearly always be accessible to the user so that they don’t get lost and can easily stop what they were previously doing.

The navigation diagram will create a visual representation of the organization of the different screens for the viewer. The corresponding buttons that move the user from one page to another will be discussed.

The wireframe will visually represent a very rough outline of the user interface, which is what the user will be seeing when navigating the application. I drew these out by hand and would like to apologize in advance for the poor drawing skills. The user interface is subject to change and is planned on being revised and updated for future versions of the system.

## **Window/Screen Navigation Diagram**

Link to diagram:

<https://drive.google.com/file/d/1K6lPFTLxSE7ETiTJ2cdSDtdUOjy4naK-/view?usp=drive_link>

A diagram of a website

Description automatically generated

## **UI Wireframes**

1. Sign-in page

A graph paper with a diagram

Description automatically generated

1. Forgot password page

A diagram of a phone

Description automatically generated

1. Account creation pageA notebook with a diagram

   Description automatically generated
2. Update information page
3. Account page

A notebook with a diagram

Description automatically generated

1. Travel plan creation page

A graph paper with a diagram

Description automatically generated

1. Search page

A sketch of a website

Description automatically generated

1. Add to plan page

A paper with writing on it

Description automatically generated

1. Modify trip plan page

A sketch of a mobile phone

Description automatically generated

1. Homepage

A graph paper with a diagram

Description automatically generated

1. Trip plan viewing page

A graph paper with a diagram

Description automatically generated

## **Reports: "Formal Output" Design**

There are no reports generated, receipts, or invoices done in the PlannerBird application. Everything will remain within the bounds of the user interface.

# **Appendices**

## **Glossary**

AWS: Amazon web services, involves cloud computing with outsourced remote servers

Budget: a budget is an estimate of intended expenses, usually restricted to a specific time

Data: information that may or may not be relevant

Deployment diagram: a diagram that shows the different relationships within the system’s architecture

Executive summary: a brief introduction to the project

Functional requirements: these are requirements of services, reactions to user input, and behavior of the software that is action-oriented

IDE: integrated development environment for programming

Non-functional requirements: these describe the characteristic requirements of a system

Resource: an important supply that provides something or someone with an asset

Risk: potential challenges or events that pose a threat to the system

Security requirements: these are company requirements that involve the security of the system

Stakeholders: parties interested in a specific project or business venture that will be affected by the development of the system

System evolution: how a system intends on grow and change in the future

UI: user interface, the application screens that the user physically interacts with and sees, front-end of the application

UML: unified modelling language, developed to standardize a way to visually diagram a system

Wireframe: a collection of visuals that roughly give an idea of what the user interface will be like for the navigation of a system

## **References / Bibliography**

Cameron, Andy (2024) In-Class Lecture notes, slides, and recordings, Seattle Pacific University.

*Free Flowchart Maker and diagrams online*. Flowchart Maker & Online Diagram Software. (n.d.). http://draw.io/

*Intelligent diagramming*. Lucidchart. (n.d.). https://www.lucidchart.com/pages/

Larman, C. (2018). *Applying UML and patterns: An introduction to object-oriented analysis and designand iterative development*. Langara College.

## **Supporting documentation**

N/A