Timelined.live

**Software Requirements Specification**

<1.0.0>

2/22/22

Uday Atragada

Ivan Bondarenko

Benji Margolis

Matt Muir

Winston Osei-Bonsu

Prepared for

CS 1530 - Software Engineering

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Description | Author | Comments |
| 02/22/22 | Initial document | Admin-Group | None |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| Signature | Printed Name | Title | Date |
|  | Benji Margolis | Software Eng. | 2/22/22 |
|  |  |  |  |
|  | Matt Muir | Software Eng. | 2/22/22 |
|  | Ivan Bondarenko | Software Eng. | 2/22/22 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Winston Osei-Bonsu | SSoftware Eng. | 2/22/22 |
|  | Uday Atragada | Software Eng. | 2/22/22 |

**Table of Contents**

REVISION HISTORY................................................................................................................................... II

DOCUMENT APPROVAL........................................................................................................................... II

1. INTRODUCTION..................................................................................................................................... 5

1.1 Purpose............................................................................................................................................. 5

1.2 Scope................................................................................................................................................ 5

1.3 Definitions, Acronyms, and Abbreviations......................................................................................... 5

2. GENERAL DESCRIPTION...................................................................................................................... 6

2.1 Product Perspective.......................................................................................................................... 6

2.2 Product Functions.............................................................................................................................. 6

2.3 Constraints, Assumptions and Dependencies................................................................................... 6

3. SPECIFIC REQUIREMENTS......................................................................................................................... 7

3.1 External Interface Requirements....................................................................................................... 7

*3.1.1 Software Interfaces.........................................................................................................................7*

3.2 Functional Requirements................................................................................................................... 9

*3.2.1 Store Data………………………………………………………............................................................................. 9*

*3.2.2 Change Data………....................................................................................................................... 9*

*3.2.3 Create Event.................................................................................................................................10*

*3.2.4 Change Event..............................................................................................................................10*

*3.2.5 Delete Event.................................................................................................................................10*

*3.2.6 Check Off Event.........................................................................................................................10*

*3.2.7 Upload Document.....................................................................................................................10*

*3.2.8 Download Document................................................................................................................10*

*3.2.9 Send Message............................................................................................................................10*

*3.2.10 Receive Message.....................................................................................................................10*

*3.2.11 Create Timeline........................................................................................................................10*

*3.2.12 Add SC to TImeline..................................................................................................................10*

3.3 Use Cases........................................................................................................................................ 11

*3.3.1 Register a New User.................................................................................................................... 12*

*3.3.2 User Login..................................................................................................................................... 14*

*3.3.3 Create Timeline.............................................................................................................................16*

*3.3.4 Generate Unique URL..................................................................................................................18*

*3.3.5 Display Dashboard.......................................................................................................................20*

*3.3.6 Generate and send email............................................................................................................20*

*3.3.7 3P is added to event...................................................................................................................20*

*3.3.8 3P modifies an event..................................................................................................................20*

*3.3.9 SC views a timeline.....................................................................................................................20*

*3.3.10 SC uploads to a timeline...........................................................................................................20*

3.4 Classes / Objects............................................................................................................................... 21

*3.4.1 DashboardBroker......................................................................................................................... 21*

*3.4.2 Timeline...........................................................................................................................................21*

*3.4.3 Event................................................................................................................................................22*

*3.4.4 Image................................................................................................................................................23*

*3.4.5 Document........................................................................................................................................23*

*3.4.6 User....................................................................................................................................................23*

*3.4.7 ServiceClient......................................................................................................................................23*

*3.4.8 ServiceProfessional............................................................................................................................23*

*3.4.9 ThirdParty............................................................................................................................................24*

3.5 Non-Functional Requirements........................................................................................................... 24

*3.5.1 Performance..................................................................................................................................... 24*

*3.5.2 Reliability..................................................................................................................................... 24*

*3.5.3 Availability................................................................................................................................... 24*

*3.5.4 Security....................................................................................................................................... 24*

*3.5.5 Maintainability............................................................................................................................. 25*

*3.5.6 Portability.................................................................................................................................... 25*

*3.4.7 User Interfaces...........................................................................................................................25*

*3.4.8 User and Human Factors........................................................................................................25*

*3.4.9 Data Retention..............................................................................................................................25*

3.7 Logical Database Requirements....................................................................................................... 25

3.8 Other Requirements.......................................................................................................................... 26

4. ANALYSIS MODELS............................................................................................................................... 26

4.1 Use Case Diagrams.......................................................................................................................... 26

4.2 Class Diagrams................................................................................................................................. 27

4.3 Sequence Diagrams.......................................................................................................................... 28

4.4 Activity Diagrams (DFD).................................................................................................................... 29

4.5 State-Transition Diagrams (STD)...................................................................................................... 30

5. CHANGE MANAGEMENT PROCESS.................................................................................................... 30

A. APPENDICES......................................................................................................................................... 31

**1. Introduction**

Timelined.live is an application made for creating timelines and schedules for long-term plans,

transactions, and otherwise. Medical providers, realtors, and other consultants will be able to

easily create a readable timeline of their course of action, contracts, and treatments for their

clients. This will allow both client and consultant to track their progress and plan for the future

in a simple yet clear and concise manner.

1.1 Purpose

In many service industries (especially healthcare) there is a big disconnect between the workers and clients when it comes to tracking where they are in a transaction or procedure. Service industries such as real estate agencies, medical offices, banking, etc. can use this application to provide a client centered solution to tackle the disconnect. Using this solution will allow service professionals to improve efficiency, be more transparent, and provide a better end product.

1.2 Scope

Timelined.live is an application geared toward making timelines for long-term projects effortless.  This web app will allow users to plan and detail projects in a way that makes sense to them.  Users will be able to keep track of data, schedule events, create displays, and appropriately format their timeline.  This application is not made for the storage of a business’s technical paperwork, such as legal documents or contracts.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 - End User - anyone who is connecting to the website server

1.3.2 - Admin - the administration team who can work on the backend parts of the application

1.3.3 - Service-Professional (SP) - a client who must be authenticated and approved by an admin, who is able to create timelines, manage service-clients, and invite service-clients

1.3.4 - Service-Client (SC) - a client who is invited to view, update, etc. a timeline from a service-professional

1.3.5 - Third-Party (3P) - an outside service professional that is allowed to view, update, and manage a part of the timeline in accordance with the needs of the SP or service client

1.3.6 - HTTPS - HyperText Transfer Protocol Secure - secure layer of communication between a web server and web browser.

1.3.7 - SSL - Secure Sockets Layer -  secure protocol developed for sending information securely over the Internet using HTTPS

1.3.8 Timeline- A plan, created by an SP and sent to SC , in which SPs can deliver details about a series of events that need to be completed for the plan to be followed through upon.

1.3.9 Event- an event is a certain step in the process according to the timeline.

**2. General Description**

Timelined.live is a solution for service professionals and service clients alike. The solution is built on the idea of using timelines to structure and plan projects in a transparent and efficient way. Service professionals can create steps for the service-clients according to their plan. Service-clients will be able to be notified, allowed to view, and updated once steps are complete.  The solution will be elegant and versatile to suit the needs of both the SP and the SC.

2.1 Product Perspective

All parties will also have dashboards in accordance to their permissions allowing them to view and modify plans. Timelined.live can be used alongside documentation products like PDFs and Microsoft Word to export data and documents. Files such as documents and photos can also be uploaded to the site for all parties to view and manage in accordance to specific needs of nodes.

2.2 Product Functions

This application will allow users to create a timeline of events that illustrate any long-term project, contract, etc. Users will be able to create events with a date and description. These events can be designed by the user according to the type of event (meetings, appointments, etc.), and will allow files to be uploaded to them. Events will also be able to be checked off by users once they are completed, which will send the user to the next event on the timeline. Users will be able to send messages to whoever else is collaborating on their timeline with them.

2.3 Constraints, Assumptions and Dependencies

The dependencies affecting the requirements listed in this SRS include a dependency on certain libraries including but not limited to, Django, djongo, MongoDB, nginx, uwsgi, and django-lockdown. Assumptions made affecting the requirements are that users can use a web browser and read English.

**3. Specific Requirements**

3.1 External Interface Requirements

3.1.1 Software Interfaces

1. Login/Sign up
   1. The purpose of this interface is to create an account that the user can login to the program with an email, or phone number, and a password
   2. The source of input for the login fields will be the user’s keyboard
   3. The valid inputs for emails must be a valid email, with a prefix consisting of letters, digits 0-9, and certain special characters, that is less than 64 characters, a domain name with an “@” symbol, and a top-level domain name such as “.com”.
   4. The valid inputs for phone numbers will be only the digits 0-9, and the input must be exactly 10 characters long.
   5. The valid inputs for passwords will include any combination of letters, numbers, and special characters that is less than 20 characters.
   6. Duplicate emails or phone numbers will be rejected.
   7. The user must choose if they are an SP, SC, or 3P from a drop-down menu.
   8. One character will be defined as one keystroke.
   9. There is no time limit for this interface.
   10. This interface will be made up of text boxes that accept input from the user’s keyboard with a submit button.
   11. The data from this interface will be one string holding the user’s email or phone number and another string holding the user’s password.
2. SP dashboard
   1. The purpose of the SP dashboard is to display all of the user’s active and past timelines, their saved timeline templates, their clients, and their account settings.
   2. The destination of the output will be the webpage for the option the user selects.
   3. The dashboard will be split into “Home”, “History”, “Clients,” and “Settings” options that will take the user to the respective webpage that is clicked on.
   4. The screen-format of this interface will be a sidebar displaying each option to the user.
3. SC dashboard
   1. The purpose of the SC dashboard is to display all of the user’s active and past timelines, and their account settings.
   2. The destination of the output will be the webpage for the option the user selects.
   3. The dashboard will be split into “Home,” “History,” and “Settings” options that will take the user to the respective webpage that is clicked on.
   4. The screen-format of this interface will be a sidebar displaying each option to the user.
4. 3P dashboard
   1. The purpose of the 3P dashboard is to display all of the user’s active and past timelines, and their account settings.
   2. The destination of the output will be the webpage for the option the user selects.
   3. The dashboard will be split into “Home,” “History,” and “Settings” options that will take the user to the respective webpage that is clicked on.
   4. The screen-format of this interface will be a sidebar displaying each option to the user.
5. Home page
   1. The purpose of the home page is to display a quick-view format of each of the user’s active timelines and notifications for events.
   2. SP users will have an additional option available to create new timelines.
   3. The destination of the output will be the webpage for the timeline selected, or to the new timeline interface.
   4. The organization of this interface will be list format.
6. History page
   1. The purpose of the history page is to display a quick-view format of each of the user’s past/completed timelines.
   2. The destination of the output will be a full-view format of the selected timeline.
   3. The organization of this interface will be list format.
7. Clients page
   1. The purpose of the clients page is to display a list of each of the user’s clients.
   2. Only SP users will be able to access this interface.
   3. The destination of the output will be the webpage of the selected client’s profile.
   4. The organization of this interface will be list format.
8. Saved templates page
   1. The purpose of the saved templates page will be to display a list of each of the user’s saved timeline templates in quick-view format.
   2. Only SP users will be able to access this interface.
   3. The destination of the output will be a full view of the template.
   4. The organization of this interface will be list format.
9. Settings page
   1. The purpose of the settings page is to show the user their account settings, and to edit it.
   2. User input will come from the user’s keyboard if the user selects to change password, phone number, or password, with the same rules as described in section 3.1.1 Interface #1.
   3. User input will come from the user’s keyboard if the user selects to change their account name or personal information.
   4. Valid inputs for a user’s name must consist of only letters, spaces, and apostrophes that must be less than 30 characters.
   5. One character will be defined as one keystroke
   6. The organization of this interface will be list format.
10. Create new timeline
    1. The purpose of this interface is to create a new timeline
    2. Users will be prompted to make a title, date, and description for the new timeline.
    3. Valid input for the title consists of anything the user enters that is less than 100 characters.
    4. Date input will be a drop down bar for the month, day, and year.
    5. Valid input for the description consists of anything the user enters that is less than 300 characters.
    6. User input will come from the user’s keyboard.
    7. One character will be defined as one keystroke.
    8. The data from this interface will be one string for the title, one string for the date, and one string for the description.
    9. The screen-format of this interface is a pop-up menu with text boxes that prompt each field.
11. Edit existing timeline
    1. The purpose of this interface is to edit existing timelines by creating, editing, and deleting events to and from the timeline.
    2. User input will come from dragging event boxes along the timeline
    3. Changes will be saved by clicking “Apply” button
    4. Events can be created, edited, deleted, and moved along timeline
12. Create new event
    1. The purpose of this event is to create a new event on a timeline
    2. Users will be prompted for a variety of event descriptors.
    3. Input source will come from the user’s keyboard when necessary
    4. User’s local files may be accessed when uploading documents or images
    5. The screen-format of this interface is a pop-up menu with text boxes, drop-down menus, and file browsers to fill in event information
13. Edit existing event
    1. The purpose of this event is to edit existing events on a timeline
    2. Users will be shown the same pop-up menu as described in Software Interfaces 3.1.1 Interface #12
    3. Users can change any of the fields or upload more documents/images to the event

**3.2 Functional Requirements**

3.2.1 Store data

Users can store data and information inside an event.

3.2.2 Change data

Users can change data and information inside an event.

3.2.3 Create event

Users can create an event that allows them to design it and add a date and description.

3.2.4 Change event

Users can make changes to an event’s design, date, description, and placement on the timeline.

3.2.5 Delete event

Users can delete an event from the timeline completely.

3.2.6 Check off event

Users can check off an event that flags it as completed.

3.2.7 Upload file

Users can upload files to an event.

3.2.8 Download file

Users can download files from an event.

3.2.9 Send message

Users can send messages to their collaborators.

3.2.10 Receive message

Users can receive messages from their collaborators.

3.2.11 Create Timeline

SPs are allowed to create Timelines according to their needs.

3.2.12 Add SC to the Timeline

SPs are allowed to add an SC to a timeline pertaining to them.

**3.3 Use Cases**

​​

|  |  |  |
| --- | --- | --- |
| Use Case ID | Primary Actor | Use Case |
| UC-3.3.1 | End User | Register a New User |
| UC-3.3.2 | End User | User Login |
| UC-3.3.3 | End User | Create Timeline |
| UC-3.3.4 | Back End | Generate Unique URLs |
| UC-3.3.5 | Back End | Display Dashboard |

3.3.1 Use Case #1 - Register a New User

​​

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID: | UC-3.3.1 | | | |
| Use Case Name: | Register a New User | | | |
| Created By: | Ivan Bondarenko | | Last Updated By: | Ivan Bondarenko |
| Date Created: | 02/18/2022 | | Last Revision Date: | 02/19/2022 |
| Actors: | | The primary actor will be the end user | | |
| Description: | | This use case describes the process for either how a service-professional (SP), service-client (SC), or third-party (3P) will create an account. | | |
| Trigger: | | The event will be triggered when the user clicks the “sign up” button on the home page or top right button. | | |
| Preconditions: | | 1. User must provide a valid email address, or phone number  2. User must be registered through the correct page | | |
| Postconditions: | | 1. User has an account created  2. User is notified by email or phone provided to confirm they provided an accurate address  3. User is asked to login  If unsuccessful:  1. User account is not created  2. User is notified of the error | | |
| Normal Flow: | | 1. User activates trigger  2. User selects whether they are SP, SC, 3P  3. User enters their desired username (email address or phone number) and desired password  4. User waits while system checks validity of entries / duplicate entries (3.1.1 Software Interface #1)  5. Email/text is sent to user to confirm account  6. Account is registered and stored to database | | |
| Alternative Flows: | | .4. User entries do not meet specified criteria   1. User is notified they do not meet criteria 2. Use case restarts and awaits another attempt   5. Message cannot be sent to specified address   1. Entry is removed from database 2. User is not able to log in with provided credentials | | |
| Exceptions: | | 1. User is already registered | | |
| Includes: | | This use case will not call other use cases | | |
| Frequency of Use: | | This use case will be executed only once per new user | | |
| Special Requirements: | | The username and password will be checked based on 3.1.1 Software Interfaces #1 to ensure no SQL injections can occur | | |
| Assumptions: | | User enters English text | | |
| Notes and Issues: | | May be adapted differently depending on SP, SC, P3 | | |

3.3.2 Use Case #2 - User Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID: | UC-3.3.2 | | | |
| Use Case Name: | User Login | | | |
| Created By: | Ivan Bondarenko | | Last Updated By: | Ivan Bondarenko |
| Date Created: | 02/18/2022 | | Last Revision Date: | 02/20/22 |
| Actors: | | The primary actor will be the end user | | |
| Description: | | This use case describes the process for how an SP, SC, or 3P will login to their created account and what they will see when logged in | | |
| Trigger: | | The event will be triggered when the user logins with their username and password on the login page | | |
| Preconditions: | | 1. User has already created an account  2. User is on the account login page | | |
| Postconditions: | | 1. User is notified that their login was a success  2. User is presented with their dashboard page  If Unsuccessful:  1. User is notified if their username or password is invalid  2. The user stays on the login page for another attempt | | |
| Normal Flow: | | 1. User enters their username  2. User enters their password  3. The system checks whether the entered username and password are valid  4. The system routes user to their dashboard page after successfully logging in | | |
| Alternative Flows: | | 3. If user does not enter any information   1. The system will alert the user that they must populate the fields 2. Use case restarts and awaits another user attempt   3. If username or password is invalid   1. Notify the user that they entered an incorrect credential 2. Use case restarts and awaits another user attempt | | |
| Exceptions: | | From step 3 of normal flow, if the database is not initialized or there is a database error   1. The login will not be completed 2. An error will be presented, alerting host machine and admins of the issue(s) 3. Use case will abort until error is corrected by host or administrator | | |
| Includes: | | This use case will call UC-3.3.5 “Display Dashboard” | | |
| Frequency of Use: | | This use case will be executed only once per user session, unless they logout and login again | | |
| Special Requirements: | | The use case must perform quickly to login a user (abiding the non-functional requirements) and will route the user type (SP, SC, 3P) to their specified dashboard | | |
| Assumptions: | | User enters English text | | |
| Notes and Issues: | | May be adapted differently depending on SP, SC, P3 | | |

3.3.3 Use Case #3 - Create Timeline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID: | UC-3.3.3 | | | |
| Use Case Name: | Create Timeline | | | |
| Created By: | Ivan Bondarenko | | Last Updated By: | Ivan Bondarenko |
| Date Created: | 02/18/2022 | | Last Revision Date: | 02/20/22 |
| Actors: | | The primary actor will be the end user (SP) | | |
| Description: | | This use case describes the process for how an SP will create a timeline (template) and invite their clients (SC) to view it | | |
| Trigger: | | The event will be triggered when the SP logs into their account and hits “+” or “create timeline” in their dashboard | | |
| Preconditions: | | 1. User is signed in  2. UC-3.3.5 Display Dashboard is called  3. User is on the account dashboard page  4. User is an SP (Service Professional) | | |
| Postconditions: | | 1. User is notified the timeline was successfully created  2. User is notified whether they want to invite SC to the created timeline  If Unsuccessful:  1. User is notified of the error(s) that occurred  2. User is redirected to dashboard home page | | |
| Normal Flow: | | 1. User activates Use Case with trigger  2. User populates timeline title, date, other relevant information to timeline  3. User adds event  4. User repeats step 3 until all events are added  5. When user is done they can save the timeline as a template (which will be saved to their dashboard) or choose to invite clients | | |
| Alternative Flows: | | 3. User adds event   1. User adds event title, date, etc. 2. User selects the event type - meeting, document upload, verification, etc. 3. User selects whether third party is required   5. Users chooses to invite clients   1. UC-3.3.4 is called to generate a unique URL for the timeline 2. The SC is emailed with the generated url from (1) above | | |
| Exceptions: | | From step 3 of normal flow, user adds too many events (maximum events per timeline = 25)   1. User is notified that they cannot add more than 25 events 2. User is prompted to continue to next step (5) | | |
| Includes: | | This use case will call UC-3.3.4 “Generate Unique URLs” and UC-3.3.5 “Display Dashboard” | | |
| Frequency of Use: | | This use case will be executed by the user every time they wish to create a timeline | | |
| Special Requirements: | | The use case must perform quickly to add events (abiding the non-functional requirements) and user must be an SP | | |
| Assumptions: | | User enters English text | | |
| Notes and Issues: | | All timelines and events will be stored as models (database records) in Django code structure | | |

3.3.4 Use Case #4 - Generate Unique URLs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID: | UC-3.3.4 | | | |
| Use Case Name: | Generate Unique URLs | | | |
| Created By: | Ivan Bondarenko | | Last Updated By: | Ivan Bondarenko |
| Date Created: | 02/18/2022 | | Last Revision Date: | 02/20/22 |
| Actors: | | The primary actor will be the back-end / host | | |
| Description: | | This use case describes the process for how the backend will generate unique URLs for timelines created by the SP and sent to the SC | | |
| Trigger: | | The event will be triggered by step 5.1 in alternative flow of UC-3.3.3 | | |
| Preconditions: | | 1. UC-3.3.3 is correctly implemented  2. Only the specified SC (from step 5 of UC-3.3.3) can view the content of the link and they must be logged in | | |
| Postconditions: | | 1. A unique valid URL is created and able to be visited by the SC  2. The intended SC receives an email containing the generated link | | |
| Normal Flow: | | 1. Use case is triggered  2. Function in models.py creates unique slug  3. Slug is registered with view in views.py  4. URL is added to urls.py  5. SP is notified that url is created  6. SC is notified by email from SC | | |
| Alternative Flows: | | 2. Unique slug   1. Unique slug could not be generated 2. SP is notified to try again   3. Slug is registered with views   1. Slug could not be registered 2. Admin team is notified 3. User is notified to try again | | |
| Exceptions: | | If a unique URL slug cannot be generated the use case will be aborted | | |
| Includes: | | This use case will call UC-3.3.6 “Generate and send email” | | |
| Frequency of Use: | | This use case will be executed every time an SP shares a timeline with a SC | | |
| Special Requirements: | | This use case will require certain python packages including random and unique slug from Django | | |
| Assumptions: | | We assume that the use case will use the latest version of Python, Django, and MongoDB | | |
| Notes and Issues: | | May need to handle certain exceptions like 404 | | |

3.3.5 Use Case #5 - Display Dashboard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID: | UC-3.3.5 | | | |
| Use Case Name: | Display Dashboard | | | |
| Created By: | Ivan Bondarenko | | Last Updated By: | Ivan Bondarenko |
| Date Created: | 02/18/2022 | | Last Revision Date: | 02/20/22 |
| Actors: | | The primary actor will be the back-end / host | | |
| Description: | | This use case describes the process for how the dashboard page will be displayed depending on the user who is logged in | | |
| Trigger: | | The event will be triggered by step 4 in normal flow of UC-3.3.2 | | |
| Preconditions: | | 1. UC-3.3.2 is correctly implemented  2. User must be of type SP, SC, or 3P | | |
| Postconditions: | | 1. A unique functional dashboard is created for the user | | |
| Normal Flow: | | 1. Use case is triggered  2. The back end will determine the type of user  3. The correct dashboard will be displayed for user type | | |
| Alternative Flows: | | 3. If user is SP   1. SP dashboard will be displayed   3. If user is SC   1. SC dashboard will be displayed   3. If user is 3P   1. 3P dashboard will be displayed | | |
| Exceptions: | | If a dashboard cannot be displayed, the use case will terminate, admins will be alerted and the user will be redirected to homepage after a 404 error | | |
| Includes: | | This use case will call all other listed use cases in some manner (with the exception of UC-3.3.1 and UC-3.3.2) | | |
| Frequency of Use: | | This use case will be executed every time a user logins or clicks the “dashboard” button | | |
| Special Requirements: | | This use case does not have any special requirements | | |
| Assumptions: | | This use case will assume 3 separate dashboards are created and can be called to serve to different user types | | |
| Notes and Issues: | | This use case will likely be a component of 3 different apps in the Django structure (SP-Dashboard, SC-Dashboard, 3P-Dashboard) | | |

3.3.6 Generate and send email

3.3.7 3P is added to event

3.3.8 3P modifies event

3.3.9 SC views timeline

3.3.10 SC uploads to timeline

**3.4 Classes / Objects**

3.4.1 DashboardBroker

3.4.1.1 Attributes

3.4.1.1.1 -userTimelines: Timeline[]

3.4.1.2 Methods

3.4.1.2.1 +getTimelineData(timeline: Timeline, user: User): String[]

3.4.1.2.2 +searchTimelines(user: User): Timeline[]

3.4.1.2.3 +createTimeline(user: User): void

3.4.1.2.4 +modifyTimeline(user: User): void

3.4.2 Timeline

3.4.2.1 Attributes

3.4.2.1.1 -events: Event[]

3.4.2.1.1 -users: User[]

3.4.2.2 Methods

3.4.2.2.1 +getEvents(): Event[]

3.4.2.2.2 +setEvents(events: Event[]): void

3.4.2.2.3 +getUsers(): User[]

3.4.2.2.4 +setUsers(users: User[]): void

3.4.2.2.5 +addEvent(): void

3.4.3 Event

3.4.3.1 Attributes

3.4.3.1.1 -status: String

3.4.3.1.2 -description: String

3.4.3.1.3 -location: String

3.4.3.1.4 -dateCreated: Date

3.4.3.1.5 -deadline: Date

3.4.3.1.6 -dateModified: Date

3.4.3.1.7 -collaborators: User[]

3.4.3.1.8 -image: Image

3.4.3.1.9 -doc: Document

3.4.3.2 Methods

3.4.3.2.1 +getStatus(): String

3.4.3.2.2 +setStatus(status: String): void

3.4.3.2.3 +getDescription(): String

3.4.3.2.4 +setDescription(description: String): void

3.4.3.2.5 +getLocation(): String

3.4.3.2.6 +setLocation(location: String): void

3.4.3.2.7 +getDateCreated(): Date

3.4.3.2.8 +getDeadline(): Date

3.4.3.2.9 +setDeadline(deadline: Date): void

3.4.3.2.10 +getDateModified(): Date

3.4.3.2.11 +setDateModified(dateModified: Date): void

3.4.3.2.12 +getCollaborators(): User[]

3.4.3.2.13 +setCollaborators(collaborators: User[]): void

3.4.3.2.14 +getImage(): Image

3.4.3.2.15 +setImage(image: Image): void

3.4.3.2.16 +getDoc(): Document

3.4.3.2.17 +setDoc(doc: Document): void

3.4.4 Image

3.4.4.1 Attributes

3.4.4.1.1 -img: JPEG

3.4.4.2 Methods

3.4.4.2.1 +getImg(): Image

3.4.4.2.2 +setImg(img: Image): void

3.4.5 Document

3.4.5.1 Attributes

3.4.5.1.1 -doc: PDF

3.4.5.2 Methods

3.4.5.2.1 +getDoc(): Document

3.4.5.2.2 +setDoc(doc: Document): void

3.4.6 User

3.4.6.1 Attributes

3.4.6.1.1 -username: String

3.4.6.1.2 -password: String

3.4.6.1.3 -email: String

3.4.6.1.4 -address: String

3.4.6.1.5 -phone: String

3.4.6.1.6 -userType: String

3.4.6.2 Methods

3.4.6.2.1 +getUsername(): String

3.4.6.2.2 +setUsername(username: String): void

3.4.6.2.3 +getPassword(): String

3.4.6.2.4 +setPassword(password: String): void

3.4.6.2.5 +getEmail(): String

3.4.6.2.6 +setEmail(email: String): void

3.4.6.2.7 +getAddress(): String

3.4.6.2.8 +setAddress(address: String): void

3.4.6.2.9 +getPhone(): String

3.4.6.2.10 +setPhone(phone: String): void

3.4.6.2.11 +getUserType(): String

3.4.6.2.12 +setUserType(userType: string): void

3.4.6.2.13 +viewTimeline(): void

3.4.7 ServiceClient

3.4.7.1 Attributes

3.4.7.1.1 -clientName: String

3.4.7.1.2 -dateOfBirth: Date

3.4.7.2 Methods

3.4.7.2.1 +getClientName(): String

3.4.7.2.2 +setClientName(clientName: String): void

3.4.7.2.3 +getDateOfBirth(): Date

3.4.7.2.8 +setDateOfBirth(dateOfBirth: Date): void

3.4.8 ServiceProfessional

3.4.8.1 Attributes

3.4.8.1.1 -professionalName: String

3.4.8.1.2 -website: String

3.4.8.2 Methods

3.4.8.2.1 +getProfessionalName(): String

3.4.8.2.2 +setProfessionalName(professionalName: String): void

3.4.8.2.3 +getWebsite(): String

3.4.8.2.4 +setWebsite(website: String): void

3.4.9 ThirdParty

3.4.9.1 Attributes

3.4.9.1.1 -thirdPartyName: String

3.4.9.1.2 -website: String

3.4.9.1.3 -category: String

3.4.9.2 Methods

3.4.9.2.1 +getThirdPartyName(): String

3.4.9.2.2 +setThirdPartyName(thirdPartyName: String): void

3.4.9.2.3 +getWebsite(): String

3.4.9.2.4 +setWebsite(website: String): void

3.4.9.2.5 +getCategory(): String

3.4.9.2.6 +setCategory(category: String): void

**3.5 Non-Functional Requirements**

3.5.1 Performance

The site should have a load time of under 5 seconds and be very responsive on various devices. We have chosen NGINX and uWSGI for hosting and handling server requests because of their emphasis on performance and speed. For narrower criteria, the runtime may be decreased to under 1 second.

3.5.2 Reliability

The system shall have proper exception handling, will be backed-up regularly, and uses trusted and proven software. Proper testing will be conducted before every version release to ensure quality and reliability.

3.5.3 Availability

The application must be available and compatible with as many devices as possible. In order to accomplish this we must create a solution which works on devices that simply need an internet connection and modern web browser.

3.5.4 Security

The backend server is secured according to modern OS-hardening best practices and principles including but not limited to: disabled root login, firewall configuration with only necessary ports and services open, limited-privileged user groups, intrusion detection and malware scanning, automatic updating and back-ups, etc. The website itself will be secured using HTTPS (SSL) and will be regularly hardened and checked for SQL injection attacks (1), DDoS attacks (2), XSS attacks (3), and MiTM attacks (4).

3.5.5 Maintainability

The project files will be hosted on Github and will be maintained by a Master-Commiter, who will ensure all changes are correct and not system breaking. Certain aspects of the site will be written as components that can be interchanged between others.

3.5.6 Portability

The system shall be easily ported to other machines that are able to connect to the internet and display a web browser. The code will be written with certain media screen breakpoints in mind, so that the website looks native across a variety of devices and screen sizes. Since all the processing will be done on the backend server, it should be compatible with many devices.

3.5.7 User Interfaces

The user interface for all clients shall prioritize elegance and ease of use. An attractive user interface allows the SP to maintain their brand image as a top tier service provider. Ease of use will allow the solution to be competitive and worthwhile against traditional methods of plan management.

3.5.8 User and Human Factors

The process of creating, viewing and modifying plans shall be easy to understand and use. Additionally, clients will have transparency within their current transaction. The platform shall also be trustable and convenient for SPs to transition their workflows into.

3.5.9 Data Retention

Timelines can be exported into formats such as PDF and Word documents such that SPs and SCs can use them for archival purposes. These documents will contain information that is important for the aforementioned archival purposes.

**3.6 Logical Database Requirements**

MongoDB will be the database running on port 27017 on the Ubuntu Server (20.0.4 LTS) and will be storing user authentication (login information), SP’s service-clients information (name, address, documents, etc.), transaction history, and any other relevant information. The data forms include string representation, integer representation, primary and foreign keys, unique dynamically-created identification numbers (IDs), email addresses, web addresses, location addresses, date and timestamp information, files, other media, and any other relevant types. The data will be retained on the host machine and will be backed up and sent to a secure cloud location daily. The integrity of the data will be protected using SHA-256 encryption provided by the MongoDB backend and will be kept secure. There will exist multiple relationships between collections, hence the use of primary and foreign keys.

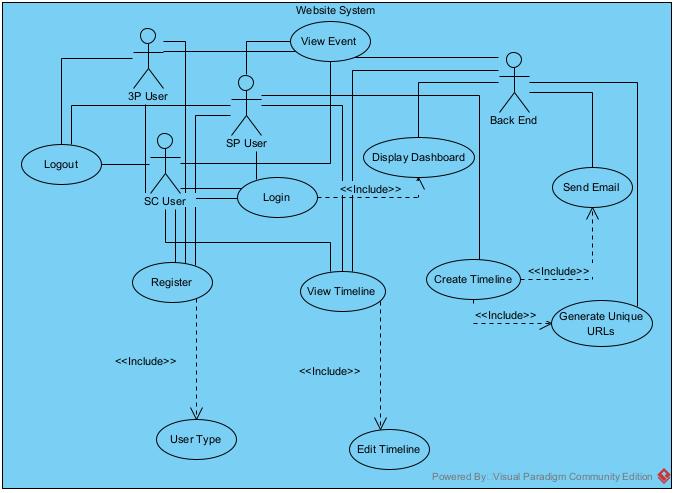
**3.7 Other Requirements**

No other requirements have yet been identified that have not been covered in the SRS.

**4. Analysis Models**

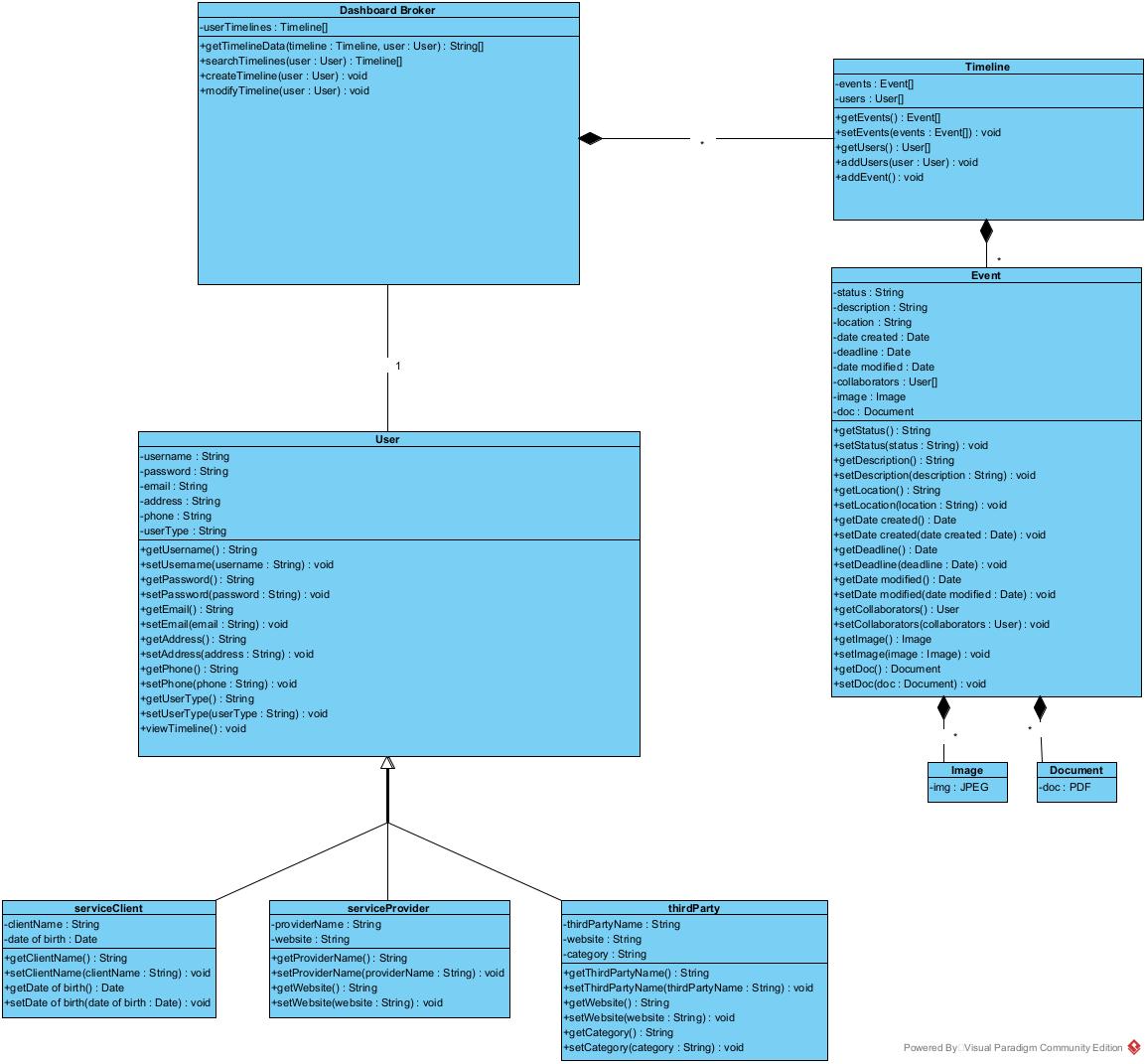
**4.1 Use Case Diagram**

This use case diagram depicts an overview of a user's interaction with our overall system. We define 4 users, the SC, SP, 3P, and back end user. The back end user deals with displaying the events, generating urls, sending emails, and administrative tasks while the other users all have similar functions, except the SC and 3P user cannot create timelines, and the 3P can only view specific events of a timeline where they are required but not the entire timeline.



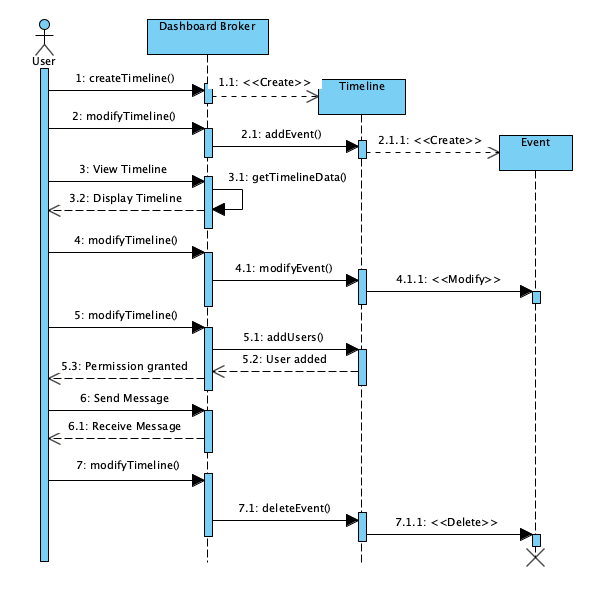
**4.2 Class Diagram**

The class diagram portrays how the nine main classes in this program will work and interact with one another. The DashboardBroker is the main class that will connect all other classes, and will be interacted with the most by the user. The User class contains information about the user, such as their name and login information. The User class is extended by three other classes, ServiceClient, ServiceProfessional, and ThirdParty, which define each possible class of user. There is also the Timeline class, which makes up an array of Event and User objects. An Event class defines one event on a Timeline, which holds all necessary information to describe that event, such as completion status, description, deadline, documents, etc. All of these classes will be implemented in Python using MongoDB as the framework.



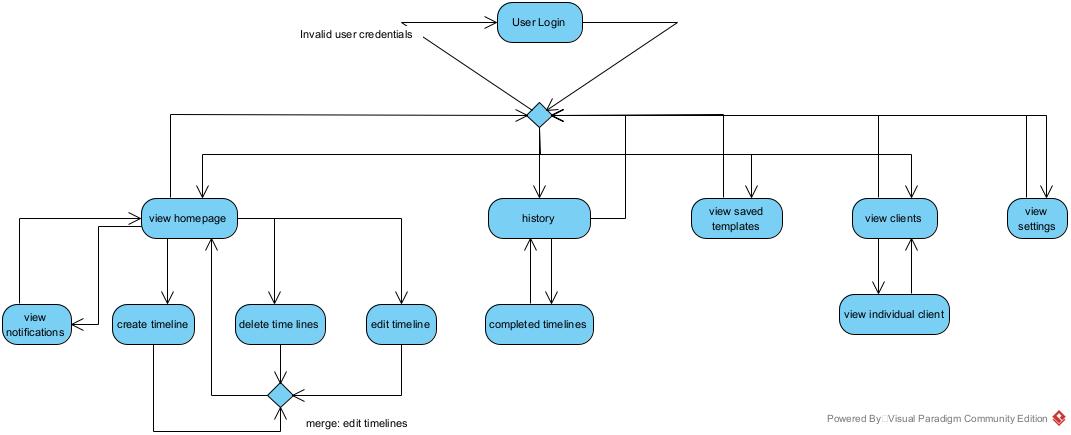
**4.3 Sequence Diagram**

The sequence diagram portrays how a general user interacts with the objects of the system.  Requests made by the user first go to the Dashboard Broker which can access the necessary methods of the timeline.  The messages below showing how the event is modified represent documents and images being uploaded, the checking-off of the event, and any attributes being modified.  For simplicity and clarity, the three types of users were combined into a generic “User” actor.  Some actions performed by this generic user can be performed by only one specific type of user.



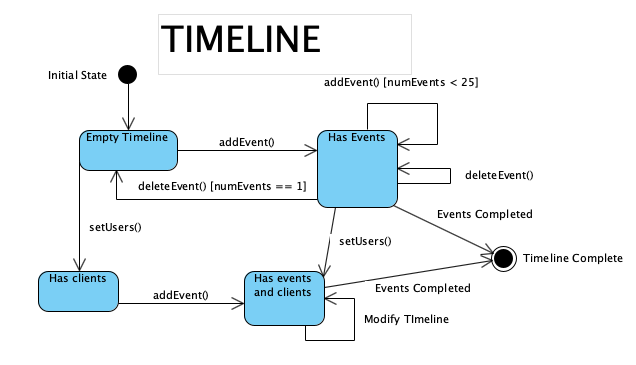
**4.4 Activity Diagram**

The activity diagram shows how the program navigates from activity to activity. The user begins at the login phase, and once the user successfully logs in, will be met with a decision to choose where to visit next in the website. They can choose among the options on the dashboard- home, history, saved templates, clients, and settings pages, which each individually lead to their own pages. The user may also choose to go back to any of the other pages listed on the dashboard at any time.



**4.5 State-Transition Diagram (STD)**

The State-Transition Diagram shows the states that a timeline can encounter after creation.  When the timeline is first created by a service provider, it contains no events and has no clients associated with it.  Clients and events are added to the timeline as needed, and a maximum number of events in a single timeline is 25.  Once all of the events in the timeline are completed, the timeline is complete.



**5. Change Management Process**

As Timelined.live is developed, this SRS may be changed and updated as functional or nonfunctional requirements are altered, added, or removed, and can be done so by any member of the development team. Each change will be discussed by the entire team and will be agreed upon before the SRS is updated.

**A. Appendices**

1. SQL Injection attacks - A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system (<https://owasp.org/www-community/attacks/SQL_Injection>)
2. DDoS (Distributed Denial of Service) attacks - A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic. (<https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>)
3. XSS (Cross-Site Scripting) attacks - Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user (<https://owasp.org/www-community/attacks/xss/>)
4. MITM attack - In cryptography and computer security, a man-in-the-middle, monster-in-the-middle, machine-in-the-middle, monkey-in-the-middle, meddler-in-the-middle (MITM) or person-in-the-middle (PITM) attack is a cyberattack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other, as the attacker has inserted themselves between the two parties (<https://en.wikipedia.org/wiki/Man-in-the-middle_attack>)