**lorelines.com**

**4. Product Requirements**

* 1. **Functional**

1. Terminology
   1. *Custom entity:* a user-created or pre-built ‘class’ in which ideas, concepts, and user creations are stored and organized.
   2. *Entity instance:* a custom entity with its fields populated with unique characteristic and identifiers
   3. *Entity schema:* the layout of *fields* in a *custom entity.*
   4. *Entity name*: the collection name for all instance of this entity.
   5. *Entity color*: the color that instances of this entity will appear as on the *timeline*. The color will default to a random, bright color for users that do not wish to maintain a *timeline*.
   6. *Instance name:* the keyword that this instance is referred to by the timeline.
   7. *Field:* Text-Field, Text-Box, Check-Box, Entity-List and/or Image
   8. *Node:* GUI objects that represent events and branching paths on the timeline
   9. *Link:* points on the left and right of *nodes* that represent ‘previous’ and ‘following’ events, visual *tethers* attach to these links.
   10. *Tether:* visual lines that connects nodes in a chronological sequence.
   11. *Instance link:* colored segments of text in *nodes* and *entity instances* that reference an *entity instance* and provide a link to view that instance.
   12. *Loreline:* The complete collection of *custom entities*, *entity instances*, and *timeline*, essentially a save file.
2. Custom Entities
   1. Ideas are stored in instances of user-created classes and pre-built classes called *custom entities*
   2. Examples of pre-built *custom entities* include: Character, City and Event. These are entities that users are likely to use and demonstrate to users how to properly construct their own *custom entities*. Pre-built entities are an optional extra when creating a new *loreline*.
   3. *Custom Entity* Creator GUI
      1. Allows user to drag and drop *fields* onto an *entity schema*
      2. *Custom entities* can be saved, edited, deleted, and duplicated
      3. When creating a new *custom entity*, the user is required to define two variables: *entity name* and *entity color.* If the entity color is left blank, it will automatically choose one for the *entity.* When creating an *entity instance* only the *instance name* must be defined.
      4. Users will be prompted by a warning when attempting to delete an entity that has one or more *entity instances*
   4. How *entity instances* are referenced on the timelineand in *entity instances*
      1. When it is determined that the user has typed an *instance name* into a *node* or an *instance field* the *instance name* will be colored the *entity color*, creating an *instance link*
      2. When the user clicks on an *instance link* the referenced instance is shown.
3. Timeline
   1. The timeline will be on a grid that will allow easy placing of *nodes* and *tethers.*
   2. Users can place *nodes* on the timeline and connect them with *tethers* to other nodes.
      1. Nodes can automatically resize to fit the amount of text in them or be a set size with a scroll wheel.
      2. *Tethers* will automatically create a straight path from one *node* to another when dragged between the two.
   3. Users can use the *instance links* feature to minimize the amount of text on the timeline.
   4. User can export timeline as an image file (possibly a premium feature).
   5. Timelines will be automatically saved to avoid loss of data.
4. Hierarchical Directory
   1. A visual way to view custom entities and entity instances without use of the timeline. Visually similar to windows file explorer.
   2. Users will be able to search for and find custom entities that they have created using this system
5. User Accounts
   1. Creating user accounts
      1. User will be asked to provide the following information when creating their account:
         1. First name
         2. Last name
         3. Username
         4. Email address
         5. Birthday (For analytics)
         6. Country (For analytics)
         7. Password
            1. Encrypted password will be stored in database
6. Creating a new *loreline*
   1. Users can create as many *lorelines* as they like (likely a premium feature)
      1. Users will have the option of importing *entities* from previous *lorelines* for cases where a new story would require pre-existing material.
7. Stretch Goals
   1. Map Maker
      1. Users can create maps to be associated with certain entities, like a city or an island
      2. User can export map as an image file (likely a premium feature).
   2. Color Themes
      1. Light theme
      2. Dark theme
   3. Ads and revenue streams
      1. Optional one-time purchase for an ad-free experience and premium features
   4. **Performance**

Lorelines will have slight load up times to due to accessing the database for user information. When the progressive web-app is loading, it will have loading screens so the consumer will know what’s happening.

* 1. **Reliability**

Lorelines should have a reliability of at least 95%. If we expect consumers to use the web-app, we need their information and storyboards as accessible as possible. Reliability is key for any storywriter that’s planning out a story.

**4.4 Data Description**

The data Lorelines will keep is each users’ custom *entities* and storyboards. Data will be kept from their first storyboards and entities to the one that their working on now. The data will primarily be text, pictures, and the storyboards themselves.

**4.5 Security and Safety**

Lorelines will hash and encrypt passwords while sending data to and from the database. Lorelines will also have to have additional security for payment if we reach our stretch goal of having ads and a premium service.

**4.6 Constraints**

Seeing as Lorelines is a sandbox storyboarding web-app, we don’t see it having to many constraints. Freedom of the consumer will be the key to the success for Lorelines.