Project Abstract

Vidol is a K-POP-related web service that helps idol fans enjoy their favorite idols' contents much more diversely. and conveniently. There are two main problems idol fans encounter when following up on K-POP content: they are so scattered and it is extremely hard to exclude unwanted parts in video content. Therefore, Vidol provides two main features: serving integrated data and extracting user-wanted parts in videos.Vidol collects and processes data of idols in advance by crawling. With the well-organized data we provide, users can follow up updates of their favorites easily. All users need to do is just typing the name of idols. It is the same in video indexing services. When users input the video URL and the idol they want to extract parts of, with machine learning, Vidol scans the video and index it either by scene change or appearing figures. With the result, users can recreate content by saving and editing it.Our goal is to boost K-POP fanship culture up by enabling efficient content consumption and recreation. As K-POP market is growing and Vidol itself promotes creating content, the importance and the utility of Vidol would get more significant over time.

Document Revision History

```
Rev. 1.0 2021-10-16 - initial version
```

Customer

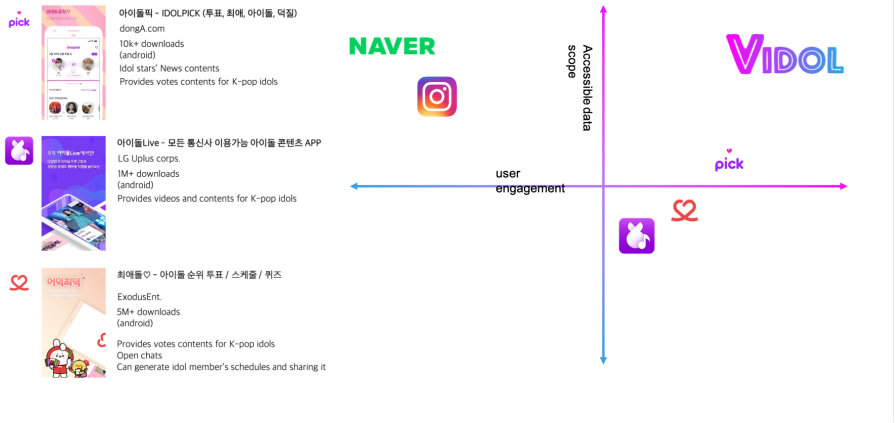
Vidol is for all who love K-Pop idols.

Specifically, Vidol can give the best user experience for those

- Tired of finding information about idols scattered around.
- Who wants to gather their favorite idols from one place.
- Who manually made the idol cut video.
- Who spends a lot of time manually finding editing points.

Competitive Landscape

According to the rapid growth of the K-pop idol market, many idol businesses and services are launched for fans of idols such as IDOLPICK, IDOL Live, and Choi-Ae-dol in south Korea. Their service provides some information about specific idols and some participatory features such as quiz or schedulers.



In our service, VIDOL, we could be a game-changer of the K-pop idol market through special features and distinctive ideas. First, VIDOL offers higher user engagement than the others. Users can leave comments on all content in VIDOL. It means that users can express their opinions free and participate in the complimentary discussion. The function of extracting video clips from the original also improves user engagement. Users who want to get their video clips from which their favorite idols come out. It could be the most famous and innovative idea in this market. Second, we manage a broad range of idol data scope by crawling from many platforms provided information on K-pop idol. Many existing services offered only a few datasets of idols or stored data manually typed. But VIDOL offers a large dataset by crawling from search engines or social media such as Naver, Instagram, and Youtube. Users who want to get information about their favorite idols can get information easily without excessive effort by searching.

User Stories

Authentification

1. Sign in

- Meta specs

Index	Content
FeatureName	User can sign in to access application
Actors	User
Precondition	User is on `Sign in Page(/login)`

- Scenario
  - **GIVEN** the user is on `Sign in Page(/login)`

  - **WHEN** the user types email and password, then click submit button
  - **THEN** if email and password are correct, the user gets redirected to `main page('/')`

- Exceptions
  - User's input has the wrong email format
  - User's input has the wrong password format
  - Wrong credential data

- Acceptance test
  - When a user clicks the submit button, loader appears and the server checks the correctness of the user's email and password
  - When the checking is finished, the user is redirected to `main page('/')`

2. Sign up

- Meta specs

Index	Content
FeatureName	User can sign up to access application (create account)
Actors	User
Precondition	User is on `Sign up Page(/join)`

- Scenario
  - **GIVEN** the user is on `Sign up Page(/login)`
  - **WHEN** the user types email, password, and name, then click submit button
  - **THEN** if all fields' inputs are correct, create account and redirect to `main page('/')`

- Exceptions
  - User's input has the wrong email format
  - User's input has the wrong password format
  - User's input has the wrong name format
  - An account with the given email already exists

- Acceptance test
  - When a user clicks the submit button, loader appears and the server checks the email address is overlapped and creates an account
  - When a user is typing the input fields, the correctness of the format appears to the user with color

Main Page

1. Search keyword

1.1 Type keyword

- Meta specs

Index	Content
FeatureName	User can type a search keyword
Actors	User
Precondition	User is on `Main Page('/')`

- Scenario
  - **GIVEN** the User is on `Main Page('/')`
  - **WHEN** the User clicks `search-input` input and types search keyword and clicks `search-button` button
  - **THEN** a list of idols related to the search keyword is displayed on the `Main page('/')`.
- Acceptance test
  - **GIVEN** the User is on `Main Page('/')` and there is 1 searh result for "IU" keyword
  - **WHEN** the User clicks `search-input` input and types "IU" and clicks `search-button` button
  - **THEN** a list of idols' length must be 1

1.2 Select idol

- Meta specs

Index	Content
FeatureName	User can click each idol that comes up as a result of a search and be redirected to `Search Result Page(/search)`
Actors	User
Precondition	User is on `Main Page('/')`, User types a search keyword and clicks `search-button` button and get results(results' length > 0)

- Scenario
  - **GIVEN** the User is on `Main Page('/')` and get search results
  - **WHEN** the User clicks one of the idols in the search results
  - **THEN** the User shuold be redirected to `Search Result`

`Page('/search')` containing idol information

- Acceptance test
  - **GIVEN** the User is on `Main Page('/')` and there are one more search results
  - **WHEN** the User clicks "IU" in the search results
  - **THEN** the User shuold be redirected to `Search Result Page('/search')` containing "IU" information

2. Ranking

2.1 Ranking in `Hottest idol tab`

- Meta specs

Index	Content
FeatureName	User can click each idol in `Hottest idol tab` and be redirected to `Search Result Page('/search')`
Actors	User
Precondition	User is on `Main Page('/')`

- Scenario
  - **GIVEN** the User is on `Main Page('/')`
  - **WHEN** the User clicks one of the idols in `Hottest idol tab`
  - **THEN** the User shuold be redirected to `Search Result Page('/search')` containing idol information
- Acceptance test
  - **GIVEN** the User is on `Main Page('/')` and there is "IU" in `Hottest idol tab`
  - **WHEN** the User clicks "IU" in `Hottest idol tab`
  - **THEN** the User shuold be redirected to `Search Result Page('/search')` containing "IU" information

2.1 Move to Ranking Page

- Meta specs

Index	Content
FeatureName	User can go to `Rank page('/rank')`
Actors	User
Precondition	User is on `Main Page('/')`

- Scenario
  - **GIVEN** the User is on `Main Page('/')`
  - **WHEN** the User clicks `Hottest idol tab`
  - **THEN** the User shuold be redirected to `Ranking Page('/rank')`
- Acceptance test
  - **GIVEN** the User is on `Main Page('/')`
  - **WHEN** the User clicks `Hottest idol tab`
  - **THEN** the User shuold be redirected to to `Ranking Page('/rank')`

Ranking Page

1. See ranking list

1.1 Ranking list

- Meta specs

Index	Content
FeatureName	User can see idol rankings
Actors	User
Precondition	User is on `Ranking Page('/rank')`

- Scenario
  - **GIVEN** the User is on `Ranking Page('/rank')`
  - **WHEN** the User on the n'th page
  - **THEN** the User can see the $10n$-$9 \sim 10$n'th idols
- Acceptance test
  - **GIVEN** the User is on `Ranking Page('/rank')`
  - **WHEN** the User clicks the 9'th page
  - **THEN** the User shuold see the 81 ~ 90'th idols

2. Move page

2.1 Move to `Search Result Page`

- Meta specs

Index	Content
FeatureName	User can click each idol in `Ranking Page('/rank')` and be redirected to `Search Result Page('/search')`
Actors	User

Index	Content
Precondition	User is on `Ranking Page('/rank')`

- Scenario
  - **GIVEN** the User is on `Ranking Page('/rank')`
  - **WHEN** the User clicks one of the idols
  - **THEN** the User shuold be redirected to `Search Result Page('/search')` containing idol information
- Acceptance test
  - **GIVEN** the User is on `Ranking Page('/rank')` and there is "IU" in ranking
  - **WHEN** the User clicks "IU"
  - **THEN** the User shuold be redirected to `Search Result Page('/search')` containing "IU" information

---

Search Result Page

1. See content of search result

1.1 Content

- Meta specs

Index	Content
FeatureName	User can see contents
Actors	User
Precondition	User is on `Search Result Page('/search')`

- Scenario
  - **GIVEN** the User is on `Search Result Page('/search')`
  - **WHEN**
  - **THEN** the User should see crawled information from Internet, SNS, Youtube and shared indexed video.
- Acceptance test
  - **GIVEN** the User is on `Search Result Page('/search')`
  - **WHEN**
  - **THEN** the User should see `info from Internet tab`, `info from SNS tab`, `info from Youtube tab`, `shared indexed video tab`.

2. Comment

2.1 Create comment

- Meta specs

Index	Content
FeatureName	User can type a comment or like comment
Actors	User
Precondition	User logged in , User is on `Search Result Page('/search')`

- Scenario
  - **GIVEN** the user is on `Search Result Page('/search')`
  - **WHEN** the user types content in `comment-input` input and clicks `comment-create` button
  - **THEN** the user's comment is added to the page.
- Exceptions
  - User inputs nothing.
  - User is not logged in.
- Acceptance test
  - When the user inputs nothing, the 'comment-create' button is disabled and gets active when the user inputs letters.
  - The comment is saved in the server after the user clicks create button and the comment is added at the top of comments.
  - When the user not logged in clicks create button, an alert message is out and the user is redirected to the login page('/login').

2.2 Like comment

- Meta specs

Index	Content
FeatureName	User can type a comment or like comment
Actors	User
Precondition	User logged in , User is on `Search Result Page('/search')`

- Scenario
  - **GIVEN** the User is on `Search Result Page('/search')`
  - **WHEN** the User clicks `comment-like` button next to a comment
  - **THEN** Likes on the comment increase by 1.

- Exceptions
  - The user already clicked like on the comment.
- Acceptance test
  - When the user clicks a button on the comment not clicked like on before, the request is handled and the number of likes increases by 1.
  - The 'comment-like' button does not appear on the comments user wrote.
  - When the user clicks a button on the comment already the user clicked like on, an alert message appears and the request is not handled.

2.3 Edit comment

- Meta specs

Index	Content
FeatureName	User can edit a comment
Actors	User
Precondition	User logged in , User is on `Search Result Page('/search')`, User is author of comment

- Scenario
  - **GIVEN** the User is on `Search Result Page('/search')` and the User is author of the comment
  - **WHEN** the User clicks `comment-edit` button next to the comment
  - **THEN** the comment becomes editable.
  - **GIVEN** the User is author of the comment and the comment is editable
  - **WHEN** the User types other content and clicks `comment-edit` button
  - **THEN** the comment is edited.
- Exceptions
  - User confirms editing with empty content.
- Acceptance test
  - When the user clicks 'comment-edit', the comment field appears with an input tag.
  - When confirming editing with empty content, an alert message appears and nothing is done.

2.4 Delete comment

- Meta specs

Index	Content
FeatureName	User can delete a comment
Actors	User
Precondition	User logged in , User is on `Search Result Page('/search')`, User is author of comment

- Scenario
  - **GIVEN** the User is on `Search Result Page('/search')` and the User is author of the comment
  - **WHEN** the User clicks `comment-delete` button next to the comment
  - **THEN** `delete-comment-confirm` pops up.
  - **GIVEN** the `delete-comment-confirm` poped up on `Search Result Page('/search')`
  - **WHEN** the User clicks `confirm` button
  - **THEN** the comment is deleted.
- Acceptance test
  - **GIVEN** the User is on `Search Result Page('/search')` and User is author of comment1
  - **WHEN** the User clicks `comment-delete` button next to the comment1
  - **THEN** confirm[Are you sure to delete this comment?] pops up.
  - **GIVEN** the `delete-comment-confirm` poped up on `Search Result Page('/search')`
  - **WHEN** the User clicks `confirm` button
  - **THEN** the comment1 is deleted.

3. Move page

3.1 Move to Video Indexing Page

- Meta specs

Index	Content
FeatureName	User can click `go-video-indexing-button` and be redirected to `Video Indexing Page('/video')`
Actors	User
Precondition	User is on `Search Result Page('/search')`

- Scenario
  - **GIVEN** the User is on `Search Result Page('/search')`
  - **WHEN** the User clicks `go-video-indexing-button` button

- **THEN** the User should be redirected to `Video Indexing Page('/video')`.

- Exceptions
  - The user is not logged in

- Acceptance test
  - When logged in user clicks the button, the user gets redirected to Video Indexing Page
  - When the user not logged in clicks the button, the user gets redirected to the login page

---

My Page

1. List of my idols

1.1 View the list

- Meta specs

Index	Content
FeatureName	List of user's favorite idol
Actors	User
Precondition	User is on `My Page('/mypage/:id')`

- Scenario
  - **GIVEN** the user is on the `My Page('/mypage/:id')`
  - **WHEN** the user scroll down to see `List of my idols`
  - **THEN** idol list of on whom the user clicked 'Like' button on the `Search Result Page('/search')`

- Exceptions
  - The user is not logged in(common exception for all features in mypage - would be omitted in later features)

- Acceptance Test
  - When the user not logged in approaches mypage, the user would be redirected to login page
  - When the user scrolls down, the list appears
  - If there is no idol the user clicked like on, the list is empty

1.2 Move to Search Result Page

- Meta specs

Index	Content
FeatureName	List of user's favorite idol
Actors	User
Precondition	User is on `My Page('/mypage/:id')` and seeing the list

- Scenario
  - **GIVEN** the user is on the `List of my idols`
  - **WHEN** the user clicks one idol
  - **THEN** user is redirected to `Search Result Page('/search')` and see specific information of the idol.
- Acceptance Test
  - When clicking one idol, loader appears and the user is redirected to Search Result Page('/search').

1.3 Cancel like

- Meta specs

Index	Content
FeatureName	List of user's favorite idol
Actors	User
Precondition	User is on `My Page('/mypage/:id')` and seeing the list

- Scenario
  - **GIVEN** the user is on the `List of my idols`
  - **WHEN** the user click 'cancel-like' button
  - **THEN** the idol user clicked gets removed from the list
- Acceptance Test
  - Loader appears right after the user clicks the cancel button and when the server successfully handles the request, the loader disappears and the idol gets removed from the list

2. Scraped articles

2.1 View scraped articles

- Meta specs

Index	Content
FeatureName	List of user's scraped articles
Actors	User
Precondition	User is on `My Page('/mypage/:id')`

- Scenario
  - **GIVEN** the user is on the `My Page('/mypage/:id')`
  - **WHEN** the user scroll down to see `Scraped articles`
  - **THEN** appear article list which scraped from the `Search Result Page('/search')`
- Acceptance Test
  - Articles appear with only titles when user scrolls down
  - When the user scrapped nothing, nothing appears

2.2 Move to Search Result Page

- Meta specs

Index	Content
FeatureName	List of user's scraped articles
Actors	User
Precondition	User is on `My Page('/mypage/:id')` and on scraped articles section

- Scenario
- **GIVEN** the user is on the `Scraped articles`
- **WHEN** the user click article name
- **THEN** user is redirected to article page in `Search Result Page('/search')` and see article of idol.
- Acceptance Test
- The loader appears when clicking the article and the user gets redirected to Search Result Page('/search').

2.3 Cancel scrap

- Meta specs

Index	Content
FeatureName	List of user's scraped articles
Actors	User
Precondition	User is on `My Page('/mypage/:id')` and on scraped articles section

- Scenario
  - **GIVEN** the user is on the `Scraped Articles`

  - **WHEN** the user click 'delete' button on one article
  - **THEN** the article gets removed from the list
- Acceptance Test
  - The loader appears when clicking delete and disappears when the server finishes handling the request

3. My Comments

3.1 View my comments

- Meta specs

Index	Content
FeatureName	List of user's comments
Actors	User
Precondition	User is on `My Page('/mypage/:id')`

- Scenario
  - **GIVEN** the user is on the `My Page('/mypage/:id')`
  - **WHEN** the user scroll down to see `My Comments`
  - **THEN** appear list of comments which is in the `Search Result Page('/search')`
- Acceptance test
  - The comments user wrote appears with the format of '[idol] comment'.
  - When no comment exists, nothing appears.

3.2 Move to Search Result Page

- Meta specs

Index	Content
FeatureName	List of user's comments
Actors	User
Precondition	User is on `My Page('/mypage/:id')`

- Scenario
  - **GIVEN** the user is on the `My Comments`
  - **WHEN** the user click comment content
  - **THEN** the user is redirected to `Search Result Page('/search')` of idol and can delete or edit their comment
- Acceptance Test

– When the user clicks a comment, the loader appears and the user gets redirected to `Search Result Page('/search')`

Video Indexing

1. Index Video by Scene Changes

1.1 Input Video by Link

- Meta specs

Index	Content
FeatureName	Input Video by Youtube Link
Actors	User
Precondition	User is on `Video Indexing Page('/video')`

- Scenario
  - **GIVEN** the user is on Video Indexing - Entry Page
  - **WHEN** the user types the link of the video to index and clicks `Cut Scenes` button
  - **THEN** the user is redirected to `Scene Cut Page`
- Exceptions
  - The link user inputs are not working.
  - The link user inputs are not from Youtube.
  - The user is not logged in(common for all features in video indexing - omitted later on)
- Acceptance test
  - When the user is not logged in, the user gets redirected to `login page('/login')`
  - 'Cut Scenes' button gets active when the user inputs a valid link
  - When a user clicks the button, the video scanning process starts and the user sees the loader until the process ends and finally gets redirected to the result page

1.2 Edit Video

- Meta specs

Index	Content
FeatureName	Edit video with the indexing
Actors	User

Index	Content
Precondition	User is on `Video Indexing Result Page('/video/result')`

- Scenario
  - **GIVEN** the user is on Video Indexing - Result Page.
  - **WHEN** the user checks videos and picks the parts to keep by clicking scenes
  - **THEN** the selected scenes are marked with colors and the total length of the selected parts appears
- Acceptance test
  - When the user clicks a scene, it gets played immediately in the player on top
  - When the user right-clicks the scene not selected, the scene gets colored
  - When the user right-clicks the scene already selected, the scene gets back to the initial state(white)

1.3 Save Video

- Meta specs

Index	Content
FeatureName	Save Video of Selected Parts
Actors	User
Precondition	User is on `Video Indexing Result Page('/video/result')`

- Scenario
  - **GIVEN** the user is on Video Indexing - Result Page
  - **WHEN** the user clicks 'Save Selected Scenes' button
  - **THEN** downloading the video of selected scenes in the user's device starts
- Exceptions
  - No scene is selected
- Acceptance test
  - When the user clicks the 'Save Selected Scenes' button, loader appears on the screen, and downloading starts when the video is ready

2. Extract Parts of Selected Idol

2.1 Input Video by Link

- Meta specs

Index	Content
FeatureName	Input Video by Youtube Link
Actors	User
Precondition	User is on `Video Indexing Page('/video')`

- Scenario
  - **GIVEN** the user is on Video Indexing - Entry Page
  - **WHEN** the user types the link of the video to index and clicks the`Extract My Idol Parts` button
  - **THEN** the user is redirected to `Search and Select Idol Page`
- Exceptions
  - The link user inputs are not working
  - The link user inputs are not from Youtube
- Acceptance test
  - 'Extract My Idol Parts' button gets active when the user inputs a valid link

2.2 Search Idol

- Meta specs

Index	Content
FeatureName	Search idol to extract parts of
Actors	User
Precondition	User submitted the link of the video

- Scenario
  - **GIVEN** the user is on Video Indexing - Search Page
  - **WHEN** the user types the keyword of idol and clicks the search button
  - **THEN** the search result with each idol's availability(of the pre-trained model) appears
- Exceptions
  - The keyword has less than 2 letters
- Acceptance test

- When the user clicks the search button, a list of idols matching the keyword appears at the bottom of the search bar

2.3 Request Support(ML training)

- Meta specs

Index	Content
FeatureName	Request training on idols Vidol doesn't have a pre-trained model of
Actors	User
Precondition	User searched idol and is seeing the search result

- Scenario
  - **GIVEN** the user searched idol and is seeing the search result
  - **WHEN** the user clicks `'Request Support'` button next to one of the idols not supported
  - **THEN** the user gets toast message 'Successfully submitted'
- Acceptance test
  - When the user clicks the `'Request Support'` button, a toast message appears and disappears a bit later

2.4 Select Idol

- Meta specs

Index	Content
FeatureName	Select idol to extract parts of
Actors	User
Precondition	User searched idol and is seeing the search result

- Scenario
  - **GIVEN** the user searched idol and is seeing the search result
  - **WHEN** the user clicks one of the idols
  - **THEN** the user is redirected to the result page
- Acceptance test
  - When the user clicks the idol, loader appears on the screen and when video processing is done, the user is redirected to the result page

2.5 Edit Video

- Meta specs

Index	Content
FeatureName	Edit video with the indexing
Actors	User
Precondition	User is on `Video Indexing Result Page('/video/result')`

- Scenario
  - **GIVEN** the user is on Video Indexing - Result Page
  - **WHEN** the user checks videos and picks the parts to keep by clicking scenes
  - **THEN** the selected scenes are marked with checkmark and total length of the selected parts appears
- Acceptance test
  - The parts with the selected idol is colored
  - When the user clicks a scene, it gets played immediately in the player on top
  - When the user right-clicks the scene not selected, a checkmark added on the scene
  - When the user right-clicks the scene already selected, the scene gets back to the initial state(no checkmark)

2.6 Save Video

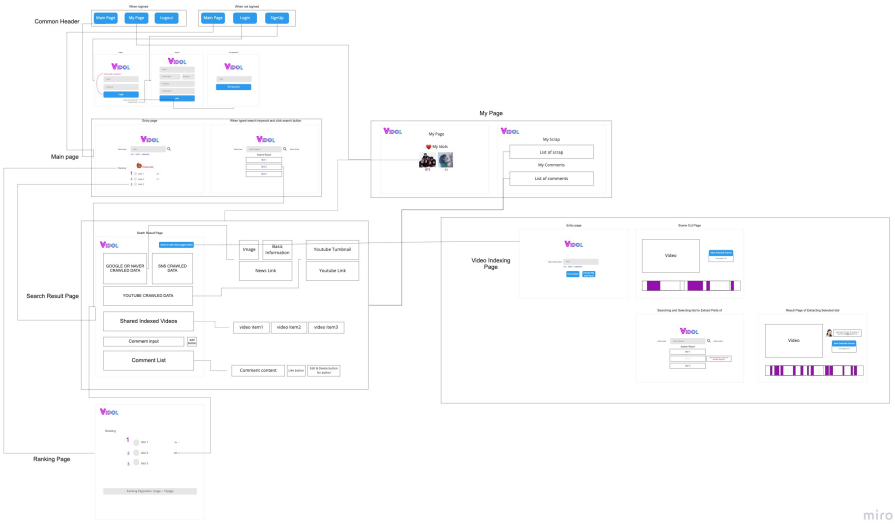
- Meta specs

Index	Content
FeatureName	Save Video of Selected Parts
Actors	User
Precondition	User is on `Video Indexing Result Page('/video/result')`

- Scenario
  - **GIVEN** the user is on Video Indexing - Result Page
  - **WHEN** the user clicks 'Save Selected Scenes' button
  - **THEN** downloading the video of selected scenes in the user's device starts.
- Exceptions
  - No scene is selected
- Acceptance test

– When the user clicks the ‘Save Selected Scenes’ button, loader appears on the screen, and downloading starts when the video is ready

User Interface Requirements



You can check the original version at: https://miro.com/app/board/o9J_lsOtDfA=/?invite_link_id=908977330