Project Name: Taskmaster

Team: Project Master

Team Member: wdong2, yuqi8, xy3, ym5, ma4, jstys

Questions:

1. Home page? (current design is fine)
2. Can requester edit accepted/bidded task?

task can be edited and deleted before assignment. also deletion is possible after finished. But when the state is "assigned" then there should be no option to delete it.

1. What is the status requirement of edit and delete? (see q2)
2. Is it possible to delete offline? (same as q2)
3. The bid price base on what? (could be any price)
4. Modify my bidding price? (before assignment)
5. Log in and password sign up? (asked)
6. Is a user name unique locally or on database? (on-line database)
7. US 05.02.01 what kind of result de we need? Tasks that I have bidded on? Or requested or bidded that is gotten from search? What is the requested tasks here?

(TA’s responce ”I think the list of bidded tasks, but I am not sure, You can get clarification from the professor.

The requirements can be changed slightly in the upcoming weeks and you can get clarification from eclass. Your design is not final. You will update and modify in the upcoming weeks according to my feedback.”)

Task basics

1.

**Use case name**: Post a new task (US 01.01.01) \*added photo and location

**Actors**:

* Task requester

**Goal**: Add task to requester task list

**Triggers**:

* Requester clicks button to add new task

**Preconditions**:

* Task requester has app
* Task requester has signed in \*
* Needs to specify title and brief description of task
* base price
* location\*?

**Postconditions**:

* Task is added to existing task list
* Task status is changed to requested

**Basic flow**:

1. Requester clicks button to add new task
2. App prompts user for information about task (title, description)
3. Requester enters the desired information
4. Requester can optionally add list of photos under 65536 bytes
5. Requester can optionally add geolocation of location of task
6. Requester clicks finish button
7. App checks information
8. Local task list for requester is updated
9. Elastic-search database is updated to include new task

**Exception**:

Exception 3: User enters incorrect input

3.1 App shows message warning the user of the error

3.2 App goes back to step 2 of basic flow

1.5

**Use case name** : View the list of requesters’ tasks (US 01.02.01)

**Actors**:

* Task requester

**Goal**: Show list of tasks that requester has requested

**Triggers**: Clicks on button to view list of my tasks

**Preconditions**:

1.The requester is registered in the app and logged in.

2. Tasks must belong to requester

3. Requester must have requested at least 1 task

**Postconditions**:

1.The requester views all his requested tasks

2.Tasks are viewable

**Basic flow**:

1. Requester clicks on ‘View my tasks’

2. App displays list of tasks that belong to the requester

**Exception**:

Exception 1: Requester has no internet connection

1.1 Displays list of local tasks

1.2 Displays warning message, saying that no connection is present

2.

**Use case name**: Edit a task (US 01.03.01) \*added photo and location

**Actors**:

* Task requester
* Pre-existing task

**Goal**: Edit an existing task

**Triggers**:

* Requester clicks an existing task

**Preconditions**:

* Need an existing task
* Task requester has signed in \*
* Task status is not done and accepted \*
* Requester clicks on task

**Postconditions**:

* Task is edited
* Local task list is changed to display updated task
* Task on elasticsearch database is updated

**Basic flow**:

1. Requester clicks on existing task
2. App shows details for that task
3. Requester clicks on ‘Edit Task’ button
4. Requester inputs changes he wants for task
5. Requester can optionally add/edit list of photos under 65536 bytes
6. Requester can optionally add/edit geolocation of location of task
7. User accepts task changes
8. App checks for errors in user input
9. Task is updated
10. Local task list is updated to show newly edited task
11. Elastic-search database is updated to include edited task when Internet is available

**Exception**:

Some unexpected actions?\*

Exception 5: User enters incorrect input

5.1 App shows message warning the user of the error

5.2 App goes back to step 4 of basic flow

Exception: User cancels edit request

* App goes back to home screen

Exception 9: Tasks can’t be saved online (user doesn’t have internet connection)

9.1 Local task list is updated

9.2 Once user regains internet connection, elastic-search database is

synced with local list

3.

**Use case name**: Delete a task (US 01.04.01)

**Actors:**

* Task requester
* Pre-existing task
* Assume some status?\*

**Goal**: Delete an existing task

**Triggers**:

* Requester clicks an existing task

**Preconditions**:

* Need an existing task
* Task requester has signed in \*
* Task status is NOT accepted \*
* Requester clicks on task

**Postconditions**:

* Task is delete
* Task list is changed to display updated task
* Task on elastic-search database if deleted

**Basic flow**:

1. Requester clicks on existing task
2. App shows details for that task
3. Requester clicks on Delete Task’ button
4. App prompts user if he really wants to delete task
5. User clicks yes/no
6. Task is deleted
7. Local task list is updated to delete task
8. Elastic-search database is updated

**Exception**:

Some unexpected actions?\*

Exception 5: User clicks ‘no’ when asked if they want to delete task

5.1 Task is not deleted

5.2 App goes back to home screen

Exception 6: Elastic-search database can’t be updated to delete task (user doesn’t have

Internet connection)

6.1 Local task list is updated to remove task

6.2 Once user regains internet connection, elastic-search database is synced

with local list

5.

**Use case name**: View details of an existing task (US 02.01.01)

**Actors**:

* Existing task
* Task requester or task provider

**Goal**: View the details of a pre-existing task

**Triggers**:

* User clicks on an existing task to view that task’s details

**Preconditions**:

* There is at least one task in task list

**Postconditions**:

* User can see the details of a given task
* Edit and delete button are shown

**Basic flow**:

1. User clicks on a task
2. Details for the task are shown in a different view

Users Profile

6.

**Use case name**: Making user profile (US 03.01.01)

**Actors**:

* Any user

**Goal**: Creates a new profile for a user

**Triggers**:

* User clicks on create new profile

**Preconditions**:

* User must be on app

**Postconditions**:

* New profile is created for user
* Data of that profile is saved to file

**Basic flow**:

1. User clicks on ‘Create new profile’
2. App displays new view, containing input request for username, contact info
3. User accepts new info
4. Profile info is saved to file
5. New profile for user is created
6. Elastic-search database if edited to include new profile

**Exceptions**:

Exception 3: Username is not unique, or username and/or contact info are not entered

3.1 App shows error message

3.2 User is prompted to re-enter correct information

7.

**Use case name**: Editing contact information from profile (US 03.02.01)

**Actors**:

* Any user

**Goal**: Edit contact information of current user

**Triggers**:

* User clicks on edit profile

**Preconditions**:

* User must have an existing profile

**Postconditions**:

* Contact information from profile is edited
* New data of that profile is saved to file

**Basic flow**:

1. User clicks on Edit profile’
2. App displays new view, containing input request for changing profile info
3. User enters new profile info
4. User accepts changes
5. Data is saved to file
6. New profile for user is created
7. Profile on elasticsearch database is edited accordingly

**Exceptions**:

Exception 3: Username is not unique, or username and/or contact info are not entered

3.1 App shows error message

3.2 User is prompted to re-enter correct information

8.

**Use case name**: Accessing another user profile (US 03.03.01)

**Actors**:

* Any user

**Goal**: Access the information of another user profile

**Triggers**:

* User clicks on an another user’s profile

**Preconditions**:

* User knows where an another user’s profile is

**Postconditions**:

* User can view the information of desired profile

**Basic flow**:

1. User clicks on another user profile
2. App displays info for that profile in a new view

Searching

9.

**Use case name**: Search for tasks (US 04.01.01 & US 04.02.01)

**Actors**:

* Task provider

**Goal**: Find all tasks with keywords entered by task provider

**Triggers**:

* Task provider enters keywords for tasks he wants to find, then searches for these tasks

**Preconditions**:

* User must be task provider

**Postconditions**:

* Task provider is shown a list of all tasks that relate to the keyword that he specified with status requested or bidded
* Task list show each task with task requester username, title, status, and lowest bid
* List is given by a series of words, separated by a space

**Basic flow**:

1. Task provider enters the search for tasks menu
2. Provider enters a list of keywords
3. Provider presses ‘Search’
4. List of tasks that relate to the list of keywords is shown to the provider
5. App allows provider to click on tasks in this list, so that he can further bid on them

**Exception**:

Exception 3: Provider enters an empty list of keywords

3.1 App causes search button to be greyed out until user enters 1 or more keywords

Task Bidding

**Use case name**: Make a bid (US 05.01.01)

**Actors**:

* Task provider

**Goal**: Provider bids on a task provided by a requester

**Triggers**:

* Send the bid price to the requester

**Preconditions**:

* The provider is registered in the app and logged in
* Task provider must be connected to the internet

**Postconditions**:

* Elastic-search database is updated
* Task requester is notified of the new bid

**Basic flow**:

1. The task provider clicks on the task that he wishes to bid on
2. App displays details for that app, as well as a prompt asking for a bid price
3. Provider enters bid price
4. Provider selects ‘Bid on task’
5. Task requester is notified on the new bid
6. Elastic-search database is updated to display new bid amount

**Exception**:

Exception 4: Provider enters incorrect bid price format

4.1 App will display error message

4.2 App goes back to step 2

Exception 6: Task was assigned while provider was bidding on it

6.1 App displays error message to provider, saying that the task is already assigned

6.2 App goes back to the search result list

**Use case name** : View the result of bidding (US 05.02.01)

**Actors**:

* Task provider
* At least 1 task must be present

**Goal**: Show list of tasks that provider has bidden on

**Triggers**: Clicks on button to view list of bidding tasks

**Preconditions**:

1.The provider is registered in the app and logged in.

2. Task must be bidded on by provider

3. Task status must be either assigned or bidded

**Postconditions**:

1. The driver sends a bid to the requester
2. On the map/history list, the task renews the status to ‘bidding’

**Basic flow**:

1. Provider sees the requests nearby or the provider searches some specific task

2. Provider chooses a task

3. System provides information of the task and ask for bidding price

4. Provider gives a bidding price and submits it.

5. System sends the bidding price to the requester and shows the lowest bidding price

6. Provider can change the bidding price based on other providers’ bidding price

**Exception**:

**Use case name**: notified of a bid on the tasks (US 05.03.01)

**Actors**:

* Task requester

**Goal**: when there is a new bid on his tasks, he should see a notification

**Triggers**:

* It will show as an apprise automatically as long as the app is still working

**Preconditions**:

* The requester has some bidding tasks
* Some task from the requester is bidded by a provider

**Postconditions**:

* The requester can know that there is a new bid

**Basic flow**:

* + - 1. The provider bid on the request’s task
      2. Elastic-search database is updated with the newest bid information
      3. The app rise the notification

**Use case name**: View my bidded tasks (US 05.04.01)

**Actors**:

* Task requester

**Goal:** Allows requester to view all his bidded tasks

**Triggers**:

* Requester presses the button to view the bidded

**Preconditions**:

* The provider is registered in the app and logged in
* Task provider must be connected to the internet

**Postconditions**:

* List of bidded tasks are shown

**Basic flow**:

1.The task provider clicks on the button to show bidded tasks

2.App displays all his bidded tasks in list

**Exception**:

Exception 2: There is no bidded tasks

2.1. Show “no bidded tasks”

**Use case name**: View bids on one task (US 05.05.01)

**Actors**:

* Task requester

**Goal:** Allows requester to view all bids on a single one his tasks

**Triggers**:

* Requester clicks on a task from the task list

**Preconditions**:

* The provider is registered in the app and logged in
* Task requester must be in the task list view

**Postconditions**:

**Basic flow**:

1. Task requester clicks on a single task from the task list (US 05.04.01)
2. App displays all bids of that task in a new view

4.

**Use case name**: Accept a bid from provider (US 05.06.01)

**Actors**:

* Registered task from requester
* Task requester

**Goal**: Accepting a bid given by a provider for a certain task

**Triggers**:

* Task requester accepts a bid from provider

**Preconditions**:

* There exists some task from requester
* There is a bid on that task
* Task status must be bidding
* internet?\*

**Postconditions**:

* Task requester pays the task provider the amount that was bid on the service
* Change task status to assigned
* Notify task provider that his bid has accepted

**Basic flow**:

1. Requester clicks on a task
2. Selects ‘Accept Bid’ option for lowest bid in task details
3. App prompts user if he wants to accept bid (yes/no popup)
4. User selects either yes or no from popup message
5. Task status is changed to assigned
6. Task provider is notified of the task assigned to him
7. Local task list is updated to task status
8. Elastic-search database is updated with status

**Exception:**

Exception 3: User presses ‘no’ on popup

3.1 App navigates user back to home screen

3.2 Task status it not changed

3.3 Task provider is not notified

**Use case name**: Decline a bid on one task (US 05.07.01)

**Actors:**

* Task requester

**Goal:** Decline a bid of one of his own task

**Triggers:**

* Requester clicks on decline button of a bid

**Preconditions:**

* The provider is registered in the app and logged in
* Task belongs to requester
* App is connected to Internet

**Postconditions:**

* The declined bid is deleted (Do we need to notify provider?)

**Basic flow:**

1.Task requester clicks on decline button

2.The bid disappears in current task

3.System updates task info on server

**Exception:**

Exception 3: Requester is offline

3.1 Display warning message

3.2 Stay on current task view

Task Assigned

4.

**Use case name**: View list of tasks that provider is assigned (US 06.01.01)

**Actors**:

* Registered task from requester
* Task provider

**Goal**: Display current tasks assigned to a task provider

**Triggers**:

* Task provider clicks on button to show all of his tasks

**Preconditions**:

* Provider has at least 1 task assigned to him
* internet?\*

**Postconditions**:

* Task provider can view all tasks assigned to him

**Basic flow:**

1. Task provider clicks on ‘View Assigned Tasks’
2. App displays list of all tasks assigned to the provider in a new view

4.

**Use case name:** View list of assigned tasks of a requester (US 06.02.01)

**Actors:**

* At least 1 assigned task from requester
* Task requester

**Goal:** Display tasks from task requester that have been assigned to providers

**Triggers:**

* Requester requests for the list

**Preconditions:**

* Requester has at least 1 task that has been assigned
* internet?\*

**Postconditions:**

* Task requester can view all tasks that have been assigned to providers
* Task list presents provider username, task title, status and bid

**Basic flow**:

1. Task requester clicks on ‘View My Assigned Tasks’
2. App displays list of all tasks from requester that have been assigned to various providers

Task Done

4.

**Use case name**: Set complete status of task (US 07.01.01)

**Actors:**

* Complete task
* Task requester

**Goal:** Set status of a complete tas

**Triggers:**

* Sets status

**Preconditions:**

* Requester must have an assigned task
* internet?\*

**Postconditions:**

* Task status turns to either ‘done’ or ‘requested’

**Basic flow:**

1. Requester opens details for a task that has been completed by provider
2. Requester sets status of task to be either done or requested
3. If set to requested, task is incomplete, task is moved back to the list so that new providers can complete it
4. If set to done, then task is complete

**Use case name**: offline behavior (US 08.01.01)

**Actors:** Any user

**Goal:** add or edit tasks while off the network, and have changes synchronized once the Internet is good

**Triggers:** add or edit tasks

**Preconditions:**

* lose internet,
* the user needs to add or edit the tasks

**Postconditions:**

* the local database/file is renewed
* when the app is online, the elastic-search database will be renewed based on the local record

**Basic flow:**

* App loses the internet
* User add or edit tasks
* App stores the record into local database
* When the internet is back, the elastic-search database will be renewed

Photographs

**Use case name**: View a photo(US 09.02.01)

**Actors**:

* Task provider
* Task requester

**Goal**: View full image of a attached photo

**Triggers**:

* Click on a photo

**Preconditions**:

* A task view is opened

**Postconditions**:

* A full image is displayed in a new page

**Basic flow**:

1.User tap a photo

2.System opens the photo in a new page

**Exceptions**: NONE

**Use case name**: Upload a photo(US 09.01.01, US 09.03.01)

**Actors**:

* Task requester

**Goal**: Attached a photo to a task

**Triggers:**

* Click on add a photo button

**Preconditions:**

* Requester has the photo to uploads

**Postconditions:**

* Valid photos are uploaded successfully

**Basic flow:**

1.User tap adding-photo button

2.System opens album to choose photos

3.Requester choose photos to be uploaded

4.Requester click on finish

5.System uploads all chosen photos

**Exceptions:**

5.check for each photo that if it is larger than equal to 65536 bytes

5.1 if true then cancel upload current one

5.2 else upload it

Geolocation

4.

**Use case name**: Set geolocation of task (US 10.01.01)

**Actors:**

* Existing task
* Task requester

**Goal:** Set geolocation to an existing task

**Triggers:**

* Clicks ‘Add Geolocation’ button

**Preconditions:**

* Must have pre-existing task
* internet?\*

**Postconditions:**

* Geolocation is assigned to selected task

**Basic flow:**

1. Requester clicks on ‘Add Geolocation’ to a new task
2. App opens new window that allows user to add new location
3. Requester enters desired location to task and accepts the location
4. App saves the location, both locally and on the elasticsearch server

**Exception:**

Exception 4: User has no internet connection

4.1 Saves changes locally

4.2 Syncs with server once internet connection has been regained

4.

**Use case name:** Display tasks of all geolocation within 5km of current position on map (US 10.03.01)

**Actors:**

* Existing task
* Task provider

**Goal:** Find all tasks on a map

**Triggers:**

* Clicks ‘Find Tasks’ button

**Preconditions:**

* Must be a registered user
* Tasks must be registered or bidded
* Task must have an assigned location
* Internet connection

**Postconditions:**

* Provider is shown a map with locations of all tasks within 5km

Basic flow:

1. Provider clicks on ‘Find tasks’ button
2. App displays map with tickers and labels for each close task

Exception:

Exception 4: User has no internet connection

4.1 App displays error message saying that connection is required

4.2 Map is not shown