Title	ユーザの写真・位置情報のグループ内シェアリングを許容したグループ内SNS
Sub Title	Inter-member social networking service allowing the sharing of group member's photograph and location information
Author	江川, 貴彦(Egawa, Takahiko) 春山, 真一郎(Haruyama, Shinichiro)
Publisher	慶應義塾大学大学院システムデザイン・マネジメント研究科
Publication year	2017
Jtitle	
JaLC DOI	
Abstract	
Notes	修士学位論文. 2017年度システムエンジニアリング学 第251号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=K040002001-00002017-0007

修士論文　　2017 年度

ユーザの写真・位置情報の
グループ内シェアリングを許容した
グループ内ＳＮＳ

江川 貴彦
(学籍番号：81534551)

指導教員　教授　春山 真一郎

2017 年 9 月

慶應義塾大学大学院システムデザイン・マネジメント研究科
システムデザイン・マネジメント専攻

論　　文　　要　　旨

学籍番号	81534551	氏　名	江川　貴彦

論 文 題 目：

ユーザの写真・位置情報のグループ内シェアリングを許容したグループ内ＳＮＳ

（内容の要旨）

スマートフォンの進化や IoT の広がりや新たなデバイスの普及などが進むにつれ，いまでは位置情報はメディアサービスだけでなく，我々の生活に大きく浸透していると感じる。しかしながらただ単に位置情報を付与しているだけで，その情報が本当に活かされてないと考えられるサービスも存在する。本研究では位置情報を活用した新しいサービスを提案したいと考えた。

既存の位置情報を活用したサービスの課題解決を検討し，イノベーティブなサービスを提案することで，関連するステークホルダの生活を微力ながら，より豊かにするような影響を与えたいと考えたのが本研究の始まりである。利用者の生活を大きく改善できるような提案に至らなくとも，既存のサービスよりユーザビリティを向上できるような提案を行うことを本研究の目的とした。

既存の位置情報を活用した位置情報ビジネスには GPS 搭載携帯電話をはじめとするモバイルデバイスを利用して利用者の位置情報を取得し，ソーシャルメディアなどのデータを活用しながら，利用者にコンテンツを配信するサービスが多数ある。既存のサービスでは，位置情報を活用しているものの，写真などのコンテンツを投稿する際は利用者が位置情報を意識して入力，もしくは，端末が表示する位置スポットを選択する必要がある。位置情報を付与する場合は利用者が意識して場所を入力や選択する必要があり，観光や旅行などで共有したい写真を投稿したい場合などに，その都度，位置情報を入力する手間があり，利用者負担の観点でユーザビリティの改善の余地がある。

本研究ではユーザビリティ向上，利用者負担の軽減，本音ベースのコニュニケーションによる信憑性の向上などを考慮し，利用者の位置と写真をグループ内でリアルタイムに共有するコミュニケーションシステムを提案した。このシステムは GPS と Beacon を利用することで，屋外だけでなく屋内でも場所と撮影した写真をスマートに共有することができる。また Instagram を活用することで，SNS のコミュニケーション特性も活かすことができ，投稿写真はユニークなハッシュタグを利用することで場所との紐づけ実現する。

提案するシステムについては主観が入らないよう，SDM 関連者の方にも実際に使って頂き，協生館や白川郷で検証を行った。その結果を元に妥当性確認を行い，提案するシステムの評価や今後の課題について検討した。

キーワード（5 語）

位置情報，グローバル・ポジショニング・システム，ビーコン，ソーシャル・ネットワーキング・サービス，ハッシュタグ

SUMMARY OF MASTER'S DISSERTATION

Student Identification Number	81534551	Name	Takahiko Egawa

Title

Inter-Member Social Networking Service Allowing the Sharing of Group Member's Photograph and Location Information

Abstract

As the evolution of smartphones, the spread of IoT and the spread of new devices progress, the position information now seems to be widely penetrated not only in media services but also in our lives. However, there are services that are considered simply that the information is merely given position information, and that information is not really utilized. In this research, I wanted to propose a new service that utilizes location information.

Thinking about solving the problem of service utilizing existing location information and thinking that we would like to influence life of relevant stakeholders with energetic but enriched effect by proposing innovative service is this research It is the beginning of. The purpose of this research was to propose a proposal that can improve usability more than existing services even though it did not lead to proposals that would greatly improve the user's life.

In the location information business utilizing existing location information, it is necessary to acquire user's location information by using a mobile device such as a GPS-equipped cellular phone and utilize data such as social media, there are many services to deliver. In existing services, although position information is utilized, when posting contents such as photographs, it is necessary for the user to input by conscious of position information or to select a position spot displayed by the terminal. When giving position information, it is necessary for the user to consciously input and select a place, and there is a time and labor for inputting position information each time, for example, when it is desired to post a picture that he / she wishes to share for sightseeing, traveling, there is room for improvement of usability from the viewpoint of user burden.

In this research, we proposed a communication system that shares users' location and photos in real time in consideration of improvement of usability, reduction of burden on users, improvement of credibility by honno based communication. By using GPS and Beacon, this system can smartly share pictures taken with places as well as indoors as well as outdoors. By utilizing Instagram, communication characteristics of SNS can also be utilized, and posted pictures realize linkage with places by using unique hashtag.

For the system we proposed, we used it by people other than me for subjectivity, and we carried out the verification at Kyoseikan and Shirakawago. We evaluated the validity based on the result and evaluated the proposed system and future issues.

Key Word(5 words)

Location Information, Global Positioning System, Beacon, Social Networking Service, Hashtag

内容

1. 序論

1.1. 研究の背景

近年のインターネットサービスには多くのものに位置情報が付与され，位置情報を扱うことそのものを目的としたサービスは特別なものではなくなってきている。スマートフォンの進化や IoT の広がりや新たなデバイスの普及などが進むにつれ，いまでは位置情報はメディアサービスだけでなく，我々の生活に大きく浸透していると感じる。しかしながらただ単に位置情報を付与しているだけで，その情報が本当に活かされてないと考えられるサービスも存在する。また位置情報は使用目的によって非常に有用であるが，扱いによっては自宅などの個人情報が漏洩する危険性もあり，使用方法について十分な配慮が必要であると感じている。これらを踏まえて，位置情報を活用したまだ世にない新しいサービスを提案したいと考えた。

1.2. 本研究の目的

既存の位置情報を活用したサービスの課題解決を検討し，イノベーティブなサービスを提案することで，関連するステークホルダの生活を微力ながら，より豊かにするような影響を与えたいと考えたのが本研究の始まりである。利用者の生活を大きく改善できるような提案に至らなくとも，既存のサービスよりユーザビリティを向上できるような提案を行うことを本研究の目的とする。

1.3. 論文の構成

2 章以降の本論文の構成を以下に示す。

2 章では 1 章で挙げた位置情報を活用した既存のサービスについて現状分析し，既存サービスの問題点を挙げ，改善策である新しいサービスを提案する。

3 章では，2 章で挙げたサービスの要求分析を行い，要求仕様を決定した。

4 章ではアーキテクチャ設計として機能設計と物理設計を行った。

5 章ではアーキテクチャ設計を元にアプリケーション実装について記載する。

6 章では提案するサービスの検証と妥当性確認を行った。

7 章では 6 章の結果より今後の課題をまとめる。

8 章では本論文の結論を記載する。

2. 現状分析と新サービスの提案

2.1. 既存のサービス

位置情報の利用は，多様な機器やサービス，技術と組み合わせて加速している。既存の位置情報を活用した位置情報ビジネスには以下のようなサービスが存在する。GPS 搭載携帯電話をはじめとするモバイルデバイスを利用して利用者の位置情報を取得し，ソーシャルメディアなどのデータを活用しながら，利用者にコンテンツを配信するサービスが多数ある。

また SNS などが提供している API を利用して，利用者の SNS 上の行動記録を得てサービスを提供している場合もある。

ビジネスに必要な位置情報の取得は GPS，Beacon や WiFi を使った測位，近距離無線通信技術を活用したものなどが存在する。

表 2-1 既存の位置情報を活用したサービスの例

#	サービス	概要	サービスの例
1.	地図サービス	地図でお店やサービス, 地域の情報を検索	Google Map, Apple Map
2.	スポット情報サービス	近くあるスポットの情報を提示する。店舗側の情報をもとに生成されるサービスと消費者サイドから生成されるサービスがある。	ぐるなび，食べログ，ホットペッパー
3.	ソーシャルメディア	投稿映像，音声，文字情報にあるコンテンツを, 当該コミュニティサービスに所属している利用者に伝え， 多数の人々が参加する双方向的な会話を実現する。	Swarm ， MOVES ， Facebook，Instagram
4.	コネクテッド・カー	自動車にインターネット通信機能を付加し，利用者の利便性を高める。	CarPlay, Android Auto, ヤフーカーナビ，ナビロー
5.	シェアード・モビリティ	自動車や自転車の使用を共有することにより, 必要な時に短距離の移動を可能にする交通・輸送手段を提供する。	全国タクシー配車，タイムズカーシェア，SMACO，ロボットタクシー
6.	オンデマンドサービス	視聴者が観たい時に様々な映像コンテンツを提供する。	Uber，Luxe，DoorDash
7.	コンテキストサービス	すぐ知りたい情報や, これから行うべき情報を前もって提供する。	Google Now, マジックハンド

2.2. 位置情報を活用した SNS の動向

近年，位置情報を活用して，スマートフォンなどから自分のいる場所，情報，コメントや写真を投稿し，他の利用者と共有するサービス群が続々と登場した。例えば，Swarm，Facebook，Instagram，Twitter，Line，Line HERE，Pinterest などが挙げられる。中でも Facebook，Instagram は主に友人とのコミュニケーションを目的に利用されることが多く，共有する情報に位置情報をタグ付けして投稿できる。

Instagram は，普段の写真投稿には位置情報を付加しない事も考えられるが，記念になる場所やみんなに伝えたい場所の場合は，位置情報を付加して投稿する利用者が多い。また Facebook も，旅行などをした際にコメント付きの写真投稿に場所をタグ付けする位置情報の使い方を好む利用者も多い。

一般の利用者は SNS では投稿に場所を付加するという，おまけ的な使い方が主流になっている。

2.3. 既存のサービスの問題点

既存のサービスで調査している範囲では，位置情報を活用しているものの，写真などのコンテンツを投稿する際は利用者が位置情報を意識して入力，もしくは，端末が表示する位置スポットを選択する必要がある。また SNS も同様で，写真投稿時に位置情報を付与することができるが，最近では写真のメタデータ(Exif)が保持している位置情報を破棄して投稿されるように仕様も変更されてきている。Instagram では写真に付加した位置情報を地図上に表示するフォットマップ機能も削除されている。

位置情報を付与する場合は利用者が意識して場所を入力や選択する必要があり，観光や旅行などで共有したい写真を投稿したい場合などに，その都度，位置情報を入力する手間がある。位置情報を付与して何十枚，何百枚以上の写真を投稿したい場合は，利用者側の負担が大きくなると考えられる。位置情報の活用したサービスは利用者の生活を豊かにする要素があると考えているが，利用者負担の観点でユーザビリティの改善の余地がある。

しかしながら，写真を投稿する際，無条件で位置情報を付与して SNS 上に公開する場合，個人情報漏洩に繋がる危険も考えられる。自宅などは位置情報を付与したくないといった要求も考えられ，利用者が意図していない位置情報付きの写真が SNS 上などで共有されないように考慮が必要である。

2.4. 新しいサービスの提案

(1) 考慮する内容と期待する効果

2.3 章の問題点を改善するために，次の点を考慮する(表 2-2)。まず，ユーザビリティの向上と利用者負担の軽減を目的に利用者の位置情報を付与する手間をなくす点を考慮する。そして，個人情報漏洩防止対策として利用者の投稿コンテンツを公開する範囲を制限する点も考慮する。後者については，公開範囲を利用者が許可しているメンバーに制限することで，より本音ベースの会話ができる効果も期待できる。その結果，コンテンツ内容の信憑性向上に繋がると考える。

また考慮する点ではプライバシーについての意思決定が一つの大切な観点だと考察する。利用者の意思でコンテンツを共有するかしないかを選択できるようにし，意図しないコンテンツが漏洩しないような仕掛けを検討する必要があると考える。

表 2-2 提案するサービスの考慮する内容と期待する効果

#	考慮する内容	期待する効果
1.	利用者の位置情報を付与する手間をなくす	ユーザビリティの向上。 利用者負担の軽減。
2.	利用者の投稿コンテンツは public ではなく，利用者が許可しているグループに制限する	個人情報漏洩防止。 本音ベースのコニュニケーションによる信憑性の向上。

(2) サービスの特徴と新規性要素の有無

表 2-2 を考慮し，SNS を活用した新しい位置情報サービスを提案する。提案するサービスは屋外だけでなく屋内で投稿したコンテンツについても考慮する。提案するサービスの特徴と新規性要素の有無について表 2-3 に示す。

#1 ～4 については既存の SNS を活用したサービスにも包括されている特徴である。一方，#5,6 については，新規性の要素を含んでいると考える。また屋外だけでなく屋内で投稿した写真についても考慮しているサービスとする。

#5 については，2.3 で挙げた通り，既存のサービスでは投稿時に利用者が位置情報を直接入力または選択する必要がある。

#6 はショッピングモールなどで，商品に近づくと商品に関連する情報をモバイル

端末などに表示させるサービスは存在するが，グループメンバーの過去の投稿内容をその場所にいくことで容易に確認できるところに新規性の要素があると考える。前者はコンテンツ提供側が予め準備している情報を伝達する手段であるが，後者はグループ内でやり取りした履歴を投稿場所にいくことで共有する手段であり，投稿コンテンツの新しい使い方だと考える。

表 2-3 提案するサービスの特徴と新規性要素の有無

#	SNS を活用したサービスの特徴	新規性要素の有無
1.	特定のグループ内のみで共有して使える	―
2.	コメントの書込みや情報の共有化を容易し，手軽に利用できる	―
3.	リアルタイムに情報伝達できる	―
4.	投稿写真に位置情報を付与する	―
5.	利用者側で投稿写真と位置情報の紐づけは不要	◎
6.	過去の投稿内容を投稿場所に行くことで容易に参照できる	○

新規性要素を含んでいると考えられる#5,6 について，実現性の観点から具体的なサービス内容を以降に示す。

(3) 新規性要素を含むサービス内容について

利用者はモバイル端末を使用して本サービスを利用する。利用者が位置情報を入力せずに，投稿写真と位置情報を紐づけする手段として GPS を使用する。GPS が届かない屋内では Beacon を用いて代替的に位置情報を付与する。GPS と Beacon 両方で位置情報が取得できる場合は，GPS を優先とする。本サービスの動的に位置情報を付与できる範囲を表 2-4 に示す。

表 2-4 提案するサービスのサポート範囲

#	位置情報の取得可否		本サービスのサポート可否
	GPS	Beacon	
1.	○	○	○
2.	○	×	○
3.	×	○	○
4.	×	×	×

取得した位置情報の緯度と経度から投稿エリアを決定し，エリアと投稿写真を紐づけする。紐づけはエリア単位にユニークなハッシュタグを作成し，エリア内で撮影された写真と対象エリアのハッシュタグと結びつけて管理する。緯度と経度を用いた投稿エリアのイメージを図 2-1 に示す。

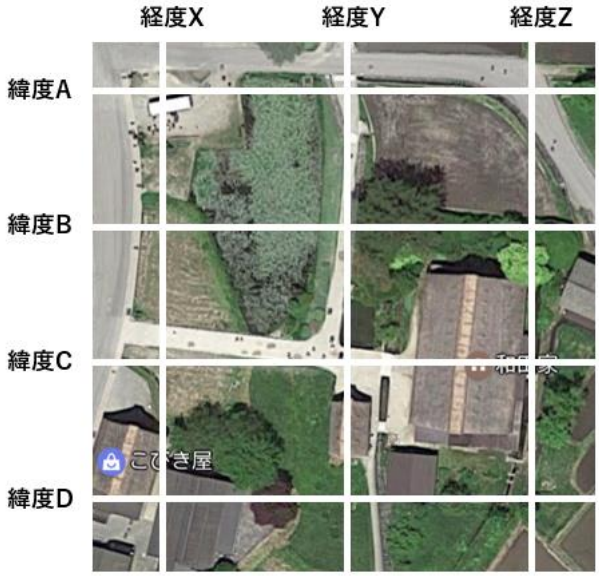


図 2-1 緯度と経度を用いた投稿エリアのイメージ 1

例えば，図 2-2 の赤枠内「投稿エリア 1」で撮影した写真を投稿した場合，付与する位置情報として「投稿エリア 1」に紐づけして，投稿コンテンツをグループ内に共有する。また，グループメンバーが「投稿エリア 1」に入った場合，グループメンバーが過去に投稿したコンテンツをモバイル端末に表示し，容易に参照できるようにする。

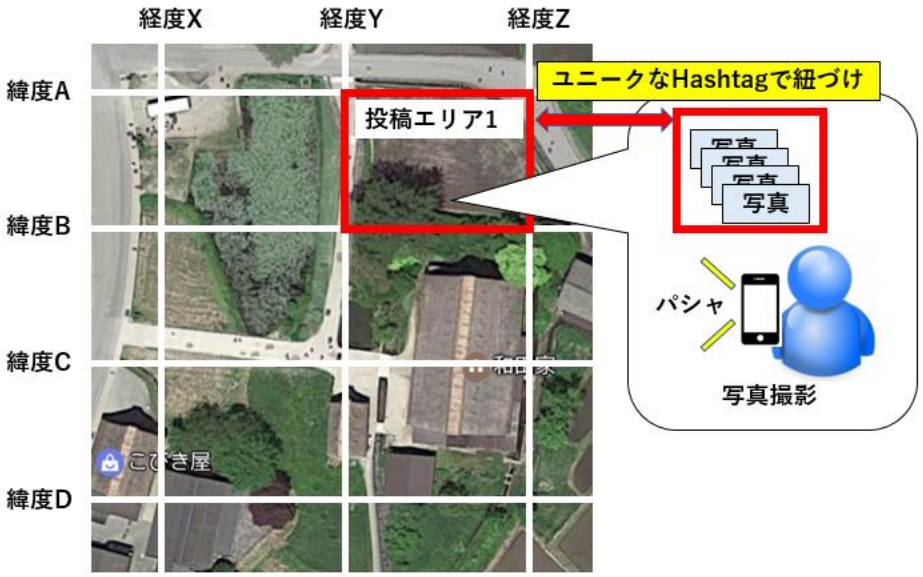


図 2-2 緯度と経度を用いた投稿エリアのイメージ 2

GPS が届かない屋内については Beacon を設置し，Beacon の UUID に対応する位置情報を予め準備することで，屋内エリアで投稿した写真と位置情報を紐づけする。図 2-3 では赤枠内 Beacon エリアで撮影した写真を投稿した場合，付与する位置情報として Beacon エリアに紐づけして，投稿コンテンツをグループ内に共有する。また，グループメンバーが Beacon エリアに入った場合，屋外同様にグループメンバーが過去に投稿したコンテンツをモバイル端末に表示し，容易に参照できるようにする。

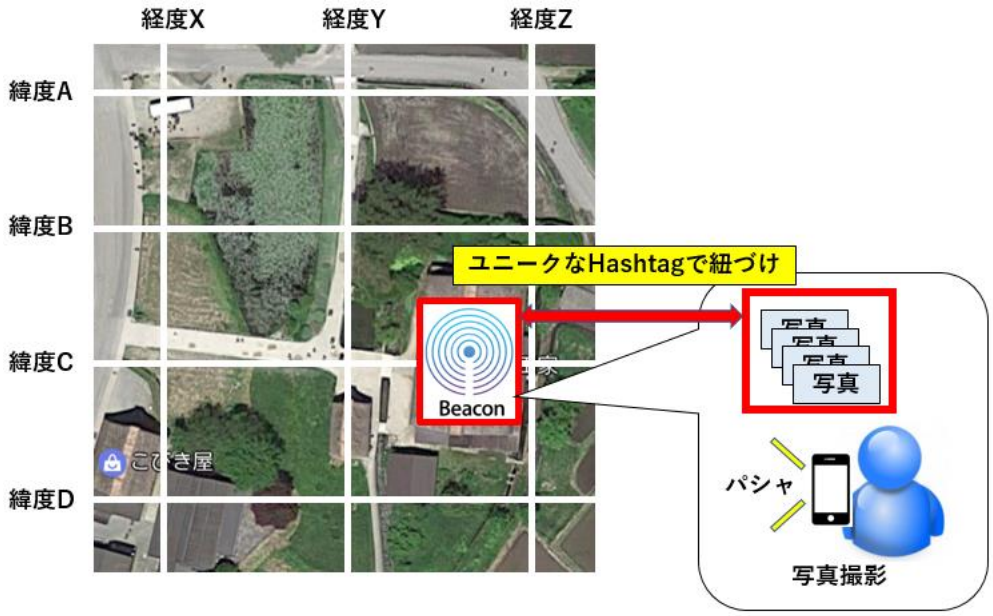


図 2-3 Beacon を用いた屋内投稿エリアイメージ

(4) 本サービスの使用例

本サービスは様々な用途で多くの使い方を提案でき，お互い許可しているメンバー間での利用を想定している。しかしながら，コンテンツを提供する側が public に情報を公開しても良いという条件を前提にできるのであれば，応用例としてより多くの使い方を提案することが可能になる。それはユーザが任意の場所に行くことで，コンテンツを容易に参照できる特性を活かすことができると考えるためである。ただし，提供するコンテンツを public に扱う場合は，個人的な情報や，第三者に知らせてはいけない情報が含まれていないかどうかの確認は必要である。情報漏洩対策の検討も踏まえ，その点を十分に配慮すべきだと考える。

任意のエリアにコンテンツを準備する応用例として，サービス提供側はその場所で撮影しなくても，該当エリアに紐づけした写真を投稿するようなアプローチも検討できる。これは事前に紐づけしたい場所の緯度・経度もしくは Beacon の UUID を把握していれば，撮影場所に依存しないでエリアに紐づく写真を投稿できるという考えである。尚，この方式は任意の場所で撮影した写真を自動で投稿する内容と趣旨が合わないと考えられるため、応用例として挙げるのみで，3 章以降では取り扱わない。

次にいくつかのサービス例を示す。

a) 旅行風景をメンバー間で共有するサービス例

ユーザの旅行履歴を写真として残し，お互い許可しているメンバー間で，そのコンテンツを共有するサービス(図 2-4)。メンバーが一度訪問した場所で写真を撮影している場合は，その内容をその場所に行くことで確認することができる。

図 2-4 旅行風景をメンバー間で共有するサービス例

b) 商品の会員情報を通知するサービス例

商品の会員に対して，商品情報等を通知するサービス(図 2-5)。お店に直接来店する会員に対して商品のお得な情報を通知し，購入意欲を促す。お店と会員双方で有益になりうる要素があり，会員はその情報を確認し必要に応じて商品を購入する。

尚，本ケースは例であり，お店側が public に情報を提供してもよいという条件であれば，会員以外の来店者に対しても商品情報を通知するサービスも検討することができると考える。

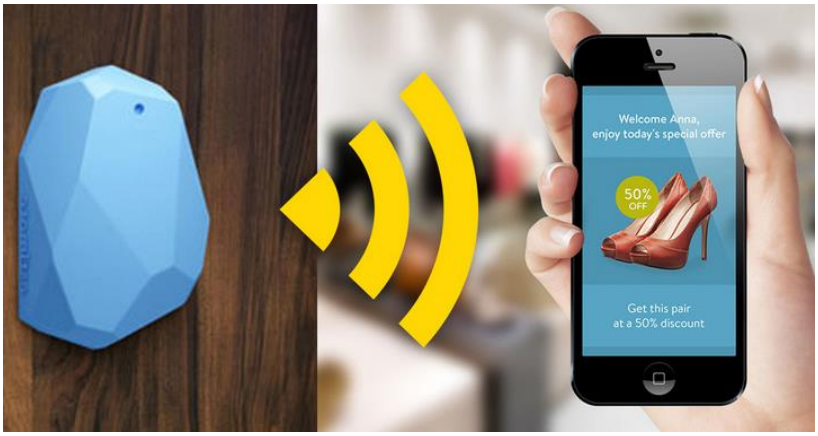


図 2-5 商品の会員情報を通知するサービス例

c) 警察の遺留品調査サービス例

事件の遺留品調査を警察関連者間で利用するサービス(図 2-6)。場所・時間・写真を紐づけして遺留品の証跡を残す。その時の遺留品の状態を確認する場合に役立つと考える。

図 2-6 警察の遺留品調査サービス例

d) 災害通知サービス例

災害が起こりうる場所で事前にその危険性を通知するサービス(図 2-7)。例えば, 通常は安全な場所であっても, 集中豪雨などのある特定の条件が整った時に, 危険になるような場所で通知するサービス。提供情報は公的な要素が強いため, 国や地方自治体等が public に通知する情報として扱える場合, 提案できるサービスだと考える。

図 2-7 災害通知サービス例

e)　危険通知サービス例

熊等が出没する危険エリアで通知するサービス(図 2-8)。山等，危険エリアが広範囲の場合，立入禁止札を全エリアに準備するのが困難な場合，代替手段として有効に利用できる要素があると考える。このケースも d)同様に公的な要素が強いため，提供範囲や提供情報を public に扱えるかどうかを十分に考慮する必要がある。投稿コンテンツは立入禁止札と違って，基本的に劣化や破損等はないと考えられるのでメンテンナスの観点でも優れている要素がある。

図 2-8 危険通知サービス例

f)　電子ポストイットサービス例

場所と覚書等を紐づけして，ユーザに通知するサービス(図 2-9)。ユーザは通知したいメッセージをポストイット等にメモをとり，それを撮影することで写真と場所の紐づけを行う。ユーザ自身が自分に対して，任意の場所で思い出したいことを事前に撮影することで，その場所に行く度に確認することができる。

図 2-9 電子ポストイットサービス例

3. 要求分析

3.1. 要求の洗出し

提案する新しいサービスは今回顧客からの要求事項ではないこともあり，本研究では，春山教授をはじめとする研究室メンバーとのミーティングや職場の方や家族などからのヒアリングをもとに要求の洗い出しを行った。その結果を表 3-1 に示す。

表 3-1 要求事項一覧

#	要求事項
1.	スマートフォンやタブレットからコンテンツを撮影しコメントや投稿ができる。
2.	撮影者の投稿操作は不要。コンテンツは動的に投稿される。
3.	誤投稿防止のため，投稿可否を確認できるようにする。
4.	投稿コンテンツは屋内，屋外問わず投稿エリア単位で集約する。
5.	投稿コンテンツはグループ内で共有する。
6.	SNS を利用したコミュニティを利用する。
7.	グループメンバーの投稿エリアに入った時に，手軽にコンテンツを参照できる。

本研究では表 3-1 の#1 の要求事項を満たすために，スマートフォンとタブレットで使用できるアプリケーションを新しいサービス提案の成果物として分析していく。

3.2. 対象システム範囲明確化

(1) システムのライフサイクル

本システムのライフサイクルを下記の図 3-1 に定義する。システムの企画フェーズから始まり，開発，運用フェーズを通じ，システムがユーザから使われなくなる破棄のフェーズで終了とする。

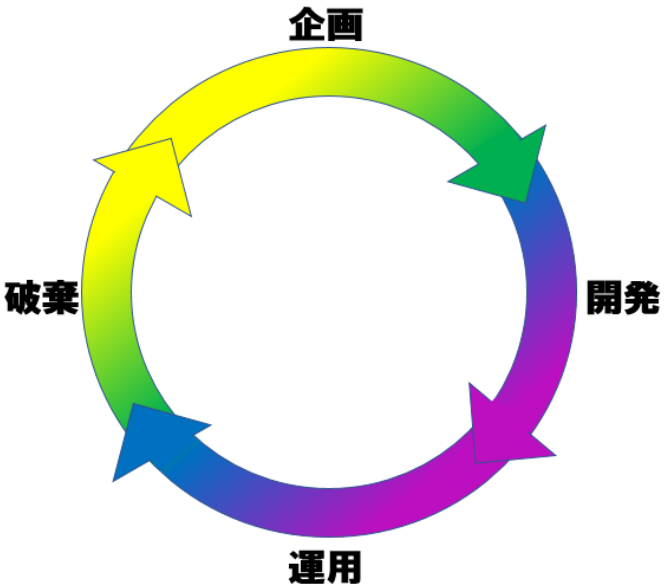


図 3-1 システムのライフサイクル

(2) コンテクスト分析

システムのライフサイクルにおいて，該当システムと外界の関係を分析する。ライフサイクルの中で利用者に実際にサービスを使用される運用フェーズをターゲットに絞り，外部から受ける影響，外部に与える影響を図 3-2 に示す。

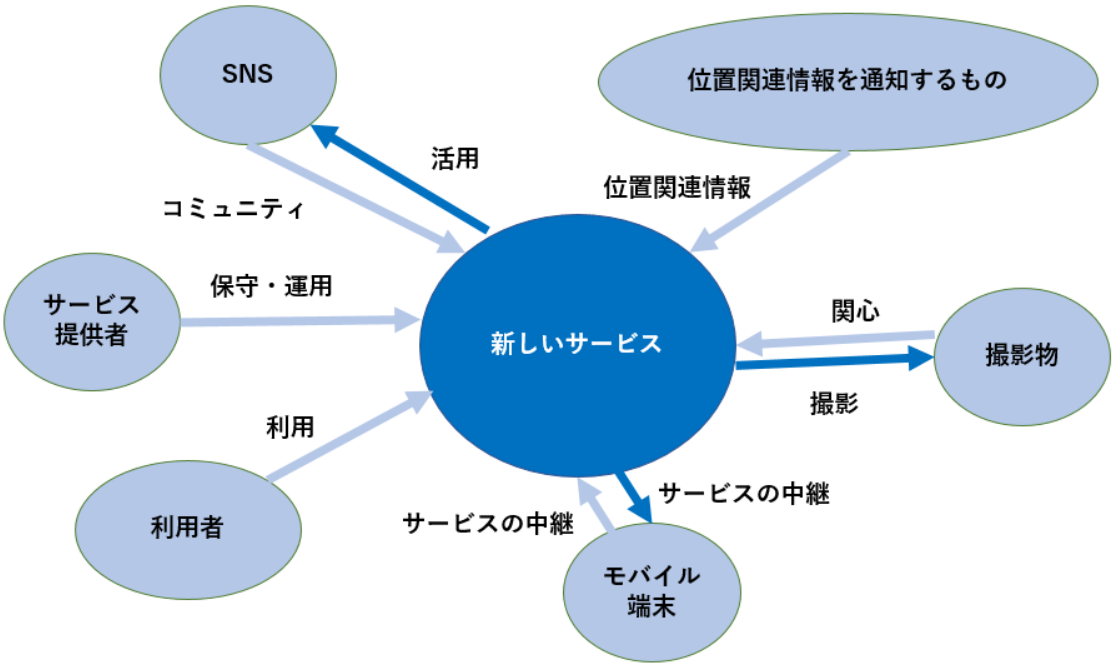


図 3-2 コンテクスト図

(3) ユースケース図

対象システム範囲をより明確化するために，システムと相互作用する外部要因を明確にする。ユースケース図を図 3-3 に示す。図 3-2 の「新しいサービス」はシステムとして，「位置関連情報を通知するもの」は屋外では GSP，屋内では Beacon として分析していく。尚，「サービス提供者」の保守・運用に関わるメンテナンスについては本論文の対象から外して分析する。

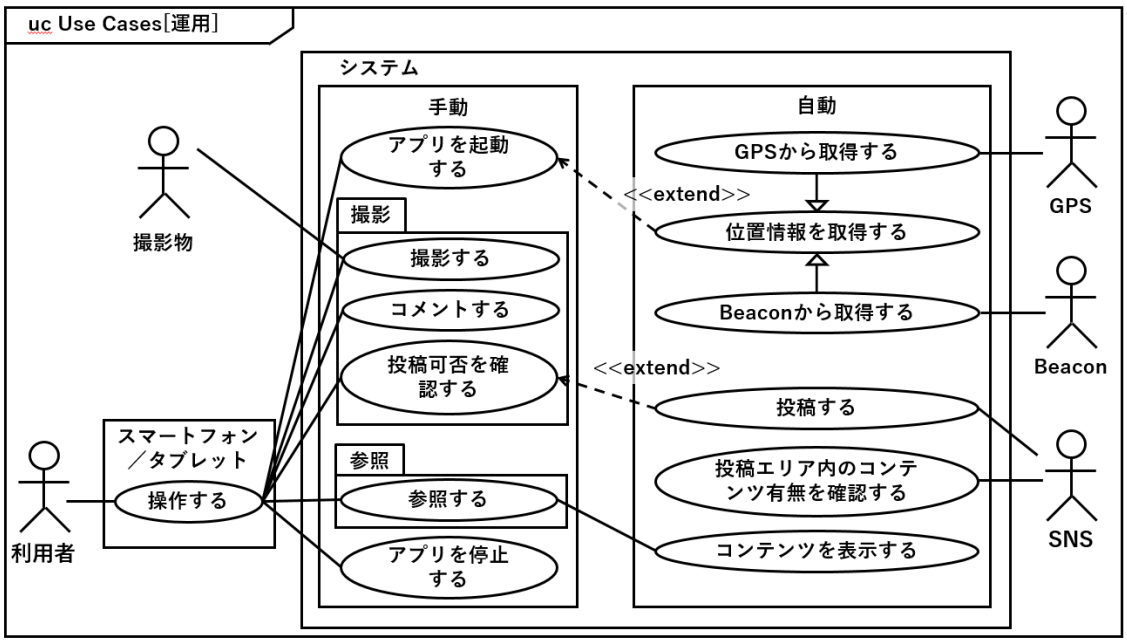


図 3-3 ユースケース図

3.3. 使用方法明確化

図 3-3 に記述したユースケースについてアクターが利用者の記述書を表 3-2～表 3-7 に示す。

表 3-2 ユースケース記述書（アプリを起動する）

#	項目	説明
1.	ユースケース名	アプリを起動する
2.	概要	本システムのアプリケーションを起動する
3.	アクター	利用者
4.	事前条件	・本システムのアプリケーションをインストール済み ・スマートフォン／タブレットがネットワークに繋がる ・位置情報サービスを ON にする ・Bluetooth を ON にする
5.	基本フロー	本システムのアイコンをタッチしてアプリを起動する
6.	代替フロー	－
7.	事後条件	バックグランドで位置情報を取得する

表 3-3 ユースケース記述書（撮影する）

#	項目	説明
1.	ユースケース名	撮影する
2.	概要	撮影物にフォーカスを合わせて撮影する
3.	アクター	利用者
4.	事前条件	・スマートフォン／タブレットがネットワークに繋がる ・位置情報サービスを ON にする ・Bluetooth を ON にする ※位置情報が取得できている場合のみ撮影可能とする。
5.	基本フロー	(1) カメラを起動する (2) 撮影物にフォーカスを合わせる (3) スマートフォン／タブレットにタッチして撮影する
6.	代替フロー	－
7.	事後条件	コメント入力ダイアログを表示する

表 3-4 ユースケース記述書（コメントする）

#	項目	説明
1.	ユースケース名	コメントする
2.	概要	写真のコメントを入力する
3.	アクター	利用者
4.	事前条件	・スマートフォン／タブレットがネットワークに繋がる ・位置情報サービスを ON にする ・Bluetooth を ON にする
5.	基本フロー	ダイアログにコメントを入力する
6.	代替フロー	－
7.	事後条件	－

表 3-5 ユースケース記述書（投稿可否を確認する）

#	項目	説明
1.	ユースケース名	投稿可否の確認をする
2.	概要	撮影した写真を投稿するかしないかを選択する
3.	アクター	利用者
4.	事前条件	・スマートフォン／タブレットがネットワークに繋がる ・位置情報サービスを ON にする ・Bluetooth を ON にする
5.	基本フロー	投稿する場合コメント入力ダイアログの OK を，投稿しない場合 Cancel をタッチする
6.	代替フロー	－
7.	事後条件	・コメント入力ダイアログを閉じる ・バックグランドでコンテンツを SNS 上に投稿する

表 3-6 ユースケース記述書（参照する）

#	項目	説明
1.	ユースケース名	参照する
2.	概要	投稿エリアのグループメンバーのコンテンツを参照する
3.	アクター	利用者
4.	事前条件	・スマートフォン／タブレットがネットワークに繋がる ・位置情報サービスを ON にする ・Bluetooth を ON にする ・グループメンバーが投稿エリアでコンテンツを投稿済み
5.	基本フロー	(1) 利用者が投稿エリアに入る (2) 投稿コンテンツがある場合，スマートフォン／タブレットのブラウザを起動し，コンテンツを表示する (3) 利用者はコンテンツを参照する (4) 必要に応じてブラウザを閉じる
6.	代替フロー	－
7.	事後条件	－

表 3-7 ユースケース記述書（アプリを停止する）

#	項目	説明
1.	ユースケース名	アプリを停止する
2.	概要	本システムのアプリケーションを停止する
3.	アクター	利用者
4.	事前条件	アプリが起動されている
5.	基本フロー	スマートフォン／タブレットの操作でアプリを終了する
6.	代替フロー	スマートフォン／タブレットの電源を OFF にする
7.	事後条件	－

3.4. 機能分析

要求を実現するための基本的なシステムの動きを図 3-4 に示す。図に各機能の識別子を割り振る。またサポート機能の範囲外の条件に対する対処を表 3-8 に示す。

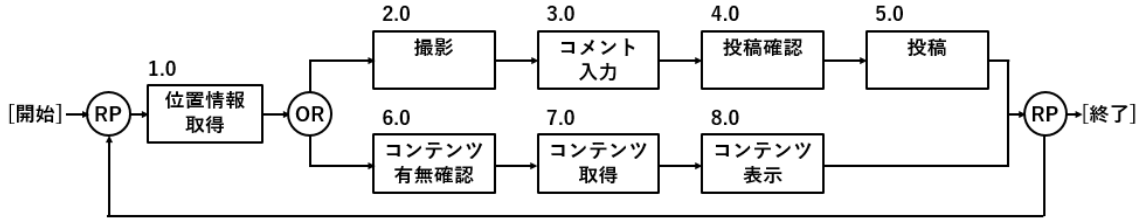


図 3-4 機能フロー図

表 3-8 範囲外動作分析表

#	範囲外条件	対処内容
1.	スマートフォン／タブレットがネットワークに繋がらない	撮影もコンテンツ表示も使用不可
2.	位置情報が取得できない場合	

3.5. 検証性識別

図 3-4 の各機能に対する試験方法概要を表 3-9 に示す。試験は実機を用いて行う。

表 3-9 テスト要求識別表

#	識別子		試験方法概要
	テスト	機能	
1.	TEST001	1.0	以下の情報が取得できるか確認する ・GPS から緯度と経度 ・Beacon の UUID
2.	TEST002	2.0	アプリでカメラを起動して撮影できるかどうか
3.	TEST003	3.0	コメントが入力できるかどうか
4.	TEST004	4.0	投稿前に OK か Cancel の選択ができるかどうか
5.	TEST005	5.0	投稿 OK を選択した後に，投稿コンテンツが SNS 上に投稿されるかどうか
6.	TEST006	6.0	エリア内に投稿済みコンテンツがあるかどうか
7.	TEST007	7.0	エリア内に投稿済みコンテンツがある場合, SNS 上からコンテンツを取得できるかどうか
8.	TEST008	8.0	投稿コンテンツが存在するエリア内に入った時にスマートフォン／タブレットに表示されるかどうか

4. アーキテクチャ設計

4.1. 機能設計

要求分析における機能分析で使用した図 3-4 により，機能の細分化を行う。それぞれ図 4-1～図 4-5 に示す。

Beacon から取得する UUID から位置情報を取得するため, 専用のサーバーを構築する。UUID に対応する位置情報とハッシュタグの管理をスマートフォン／タブレット側ではなく，サーバー側で行う設計方針とする。Beacon の追加時などに UUID 管理台帳のメンテナンスを行うが，UUID をサーバー側で管理することで，管理台帳の更新毎にスマートフォン／タブレット側のアプリケーションをアップグレードする必要がなくなる。そのため，利用者側の負担を軽減できる要素があると考える。しかしながら，今回構築するサーバーでは Beacon の UUID を位置情報や位置スポットへ変換することは行わない。サーバー側で Beacon から取得した UUID を位置情報や位置スポットが分かる名称に変換する方式は今後の課題として，今回は UUID をそのまま SNS 上に登録するハッシュタグとして利用する。

尚，投稿コンテンツの管理をサーバー側に集約したいため，コンテンツの投稿はスマートフォン／タブレット側から一旦サーバー側に送信し，サーバー側から SNS 上に投稿する設計とする。投稿済みコンテンツの取得も一旦サーバー側から SNS から取得し，コンテンツをスマートフォン／タブレット側に送信し，ブラウザで表示する流れとする。

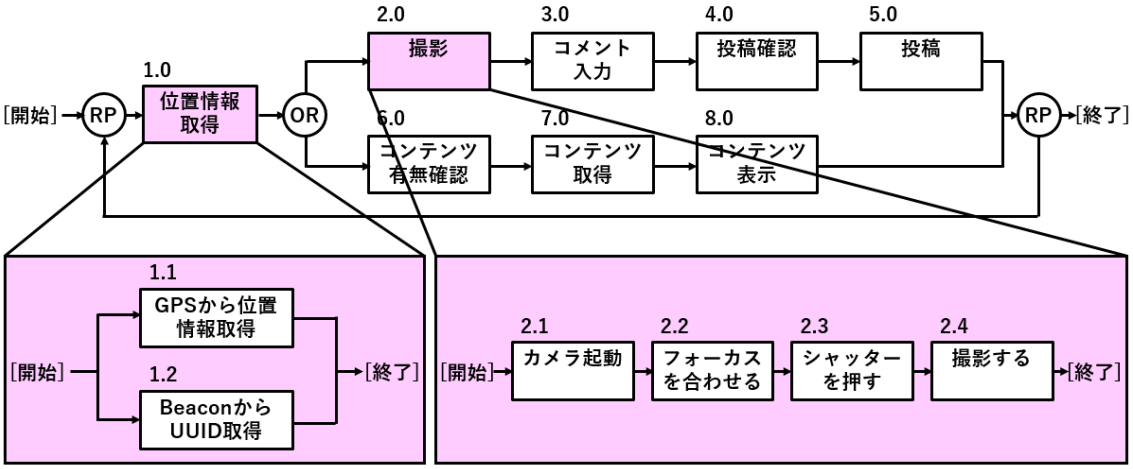


図 4-1 位置情報取得と撮影機能の細分化

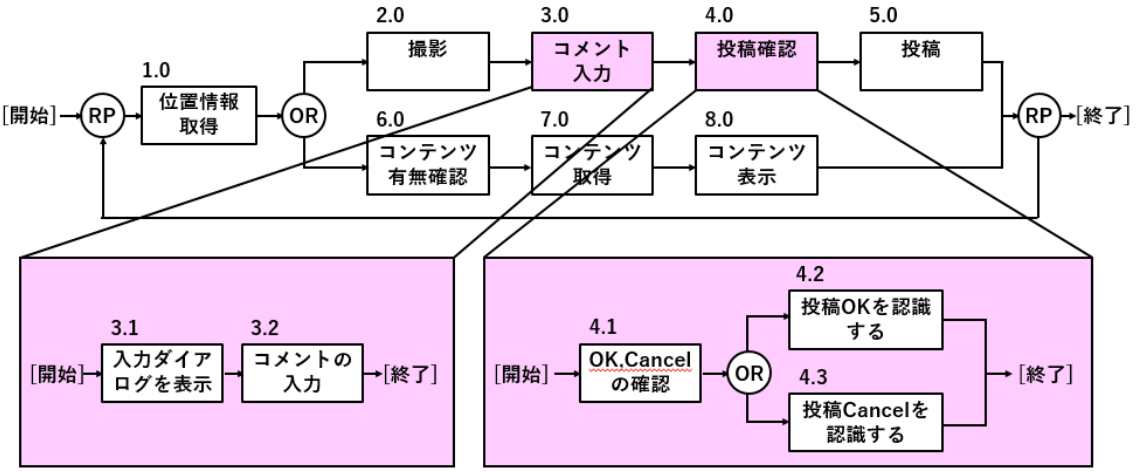


図 4-2 コメント入力と投稿確認機能の細分化

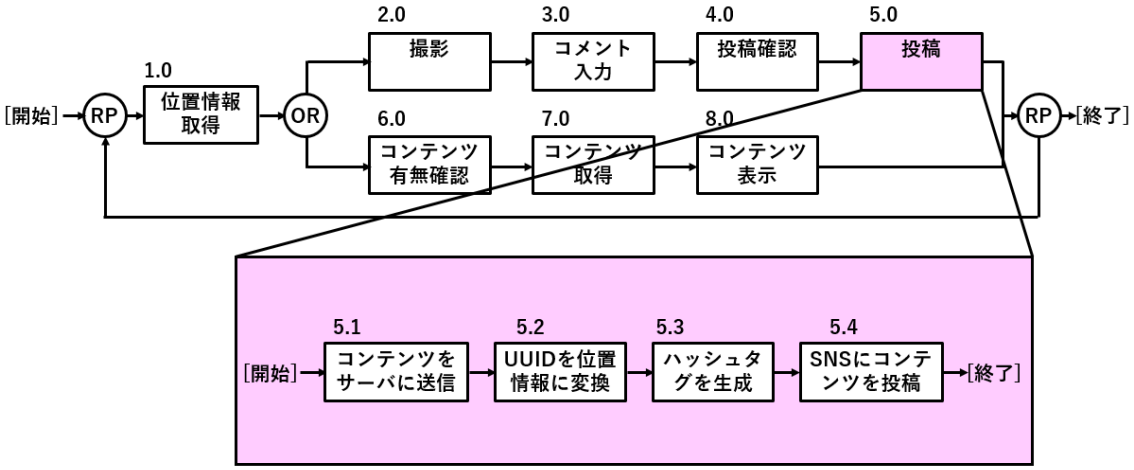


図 4-3 投稿機能の細分化

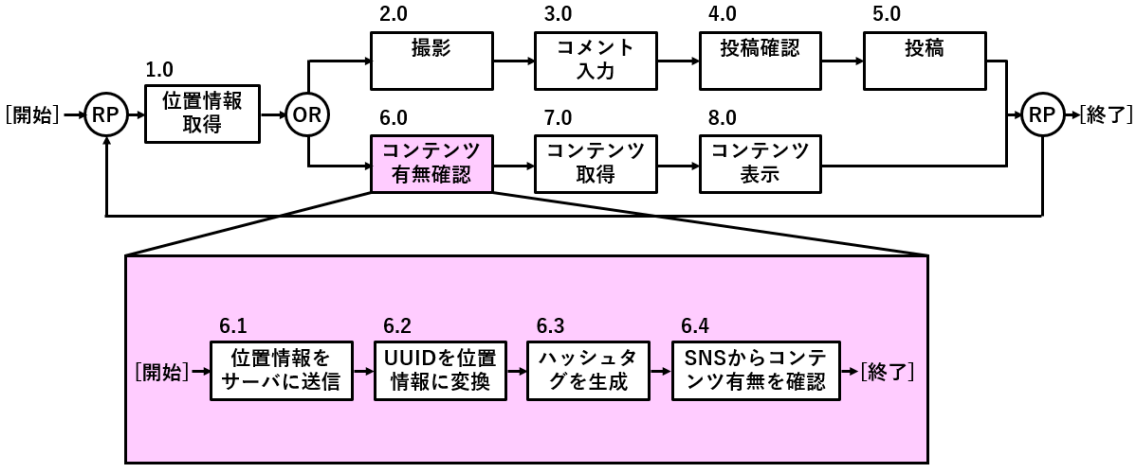


図 4-4 コンテンツ有無確認機能の細分化

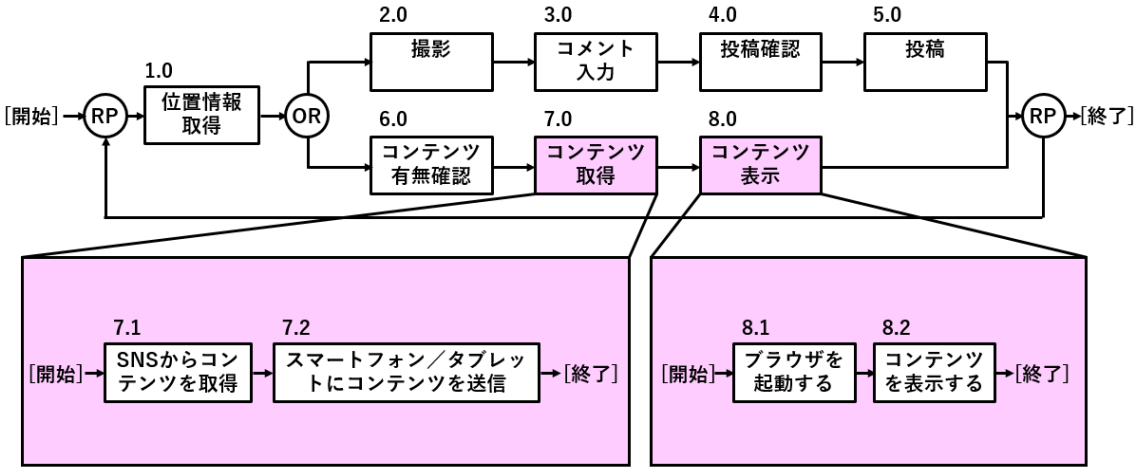


図 4-5 コンテンツ取得とコンテンツ表示機能の細分化

4.2. 物理設計

システムの物理階層構造を図 4-6 に示す。

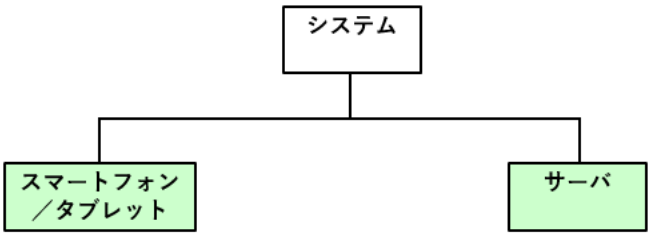


図 4-6 システムの物理階層構造

機能設計にて細分化した機能の物理階層への割振を図 4-7～図 4-11 に示す。

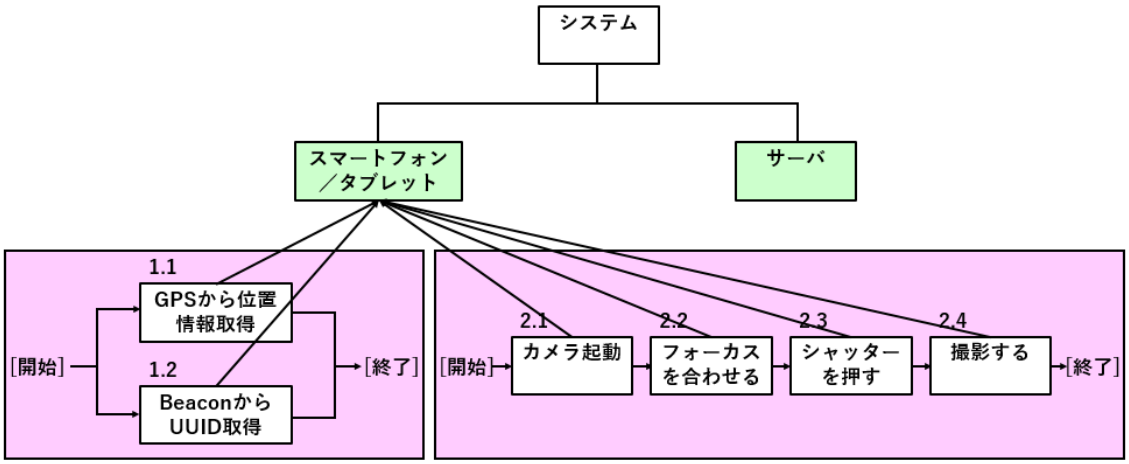


図 4-7 位置情報取得と撮影機能の物理階層への割振り

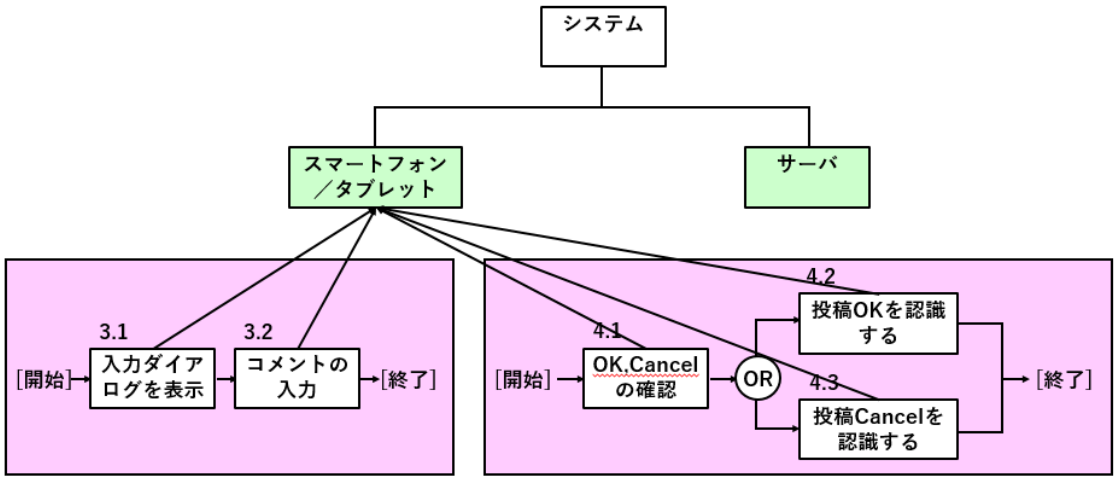


図 4-8 コメント入力と投稿確認機能の物理階層への割振り

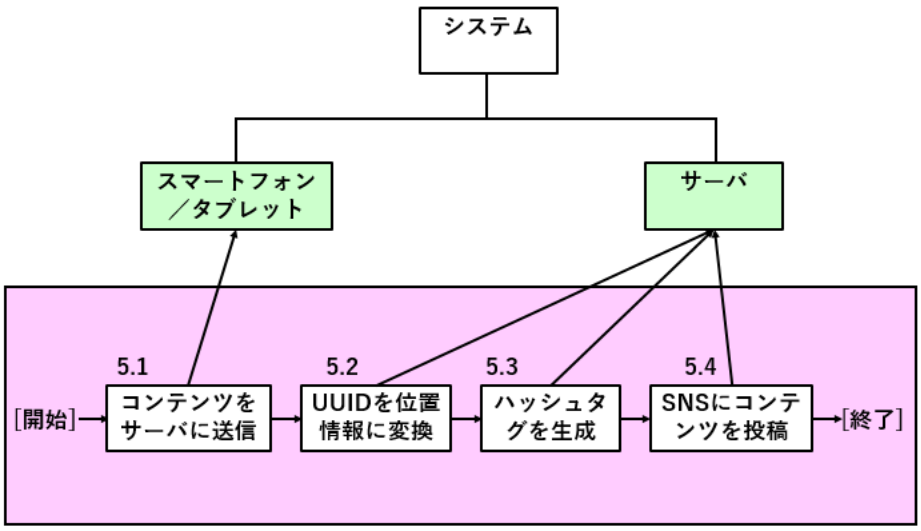


図 4-9 投稿機能の物理階層への割振り

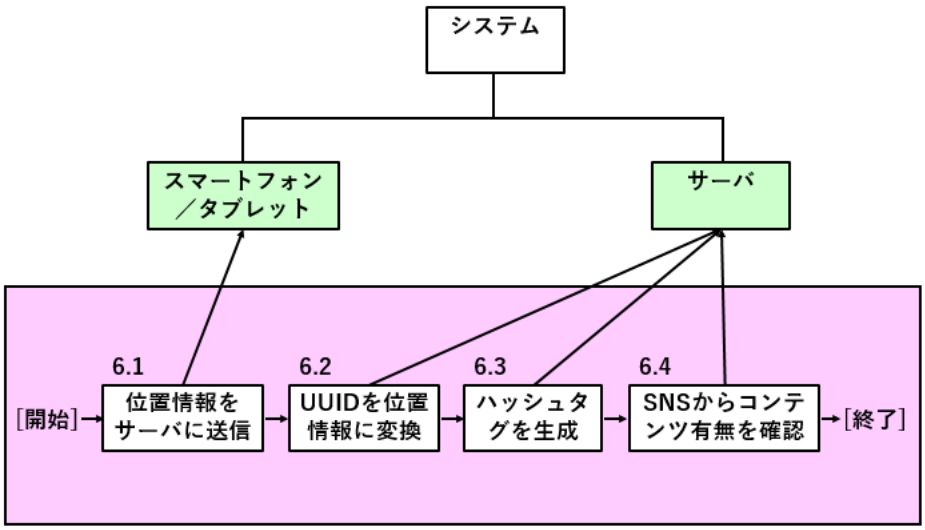


図 4-10 コンテンツ有無確認機能の物理階層への割振り

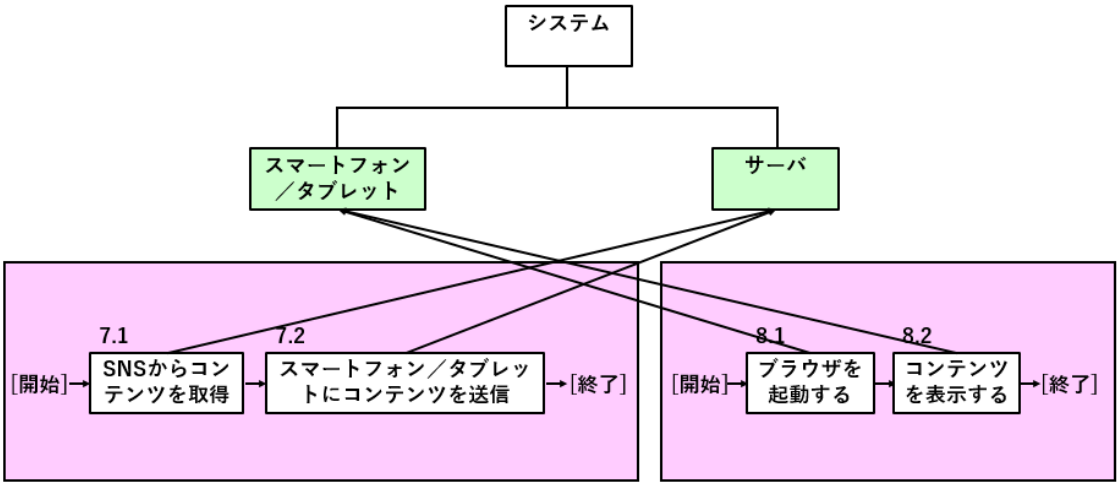


図 4-11 コンテンツ取得とコンテンツ表示機能の物理階層への割振り

機能の物理階層への割振の結果を元に，撮影と参照のシステムをそれぞれ図 4-12 と図 4-13 のブロック図に示す。

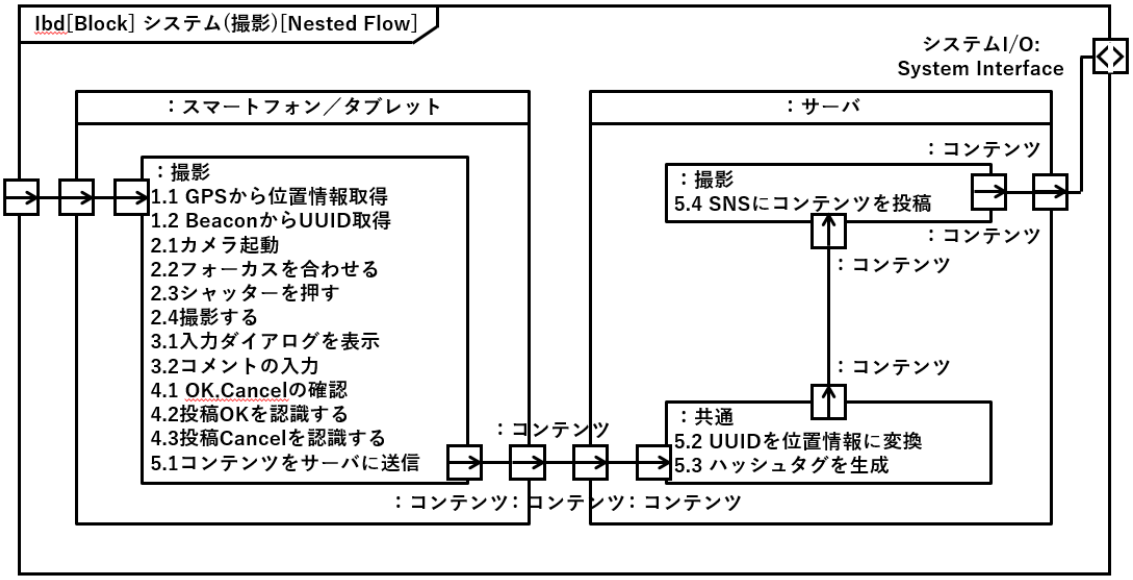


図 4-12　システム(撮影)のブロック図

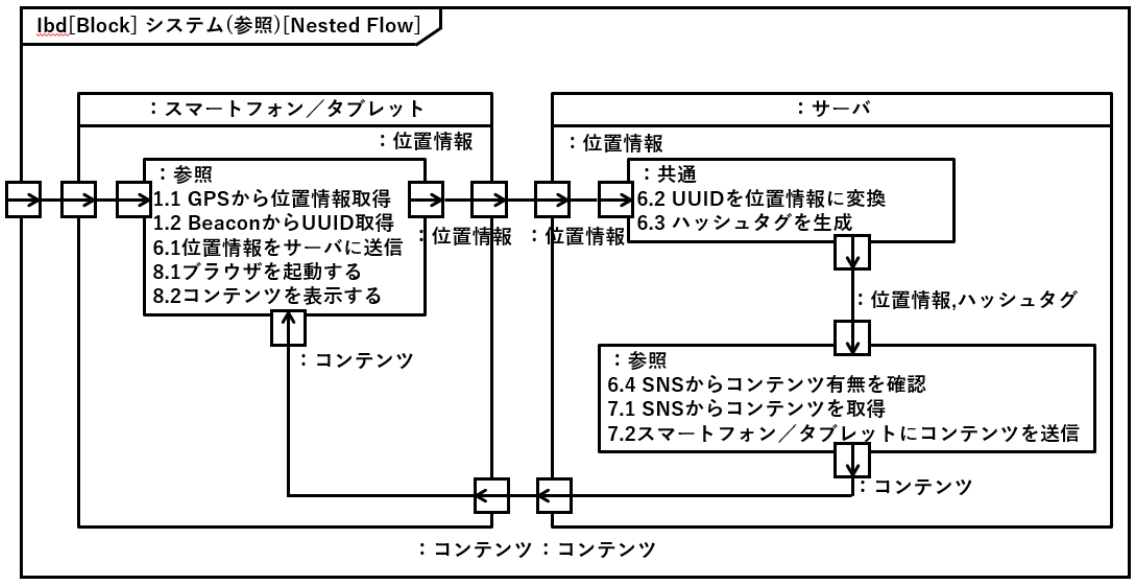


図 4-13 システム(参照)のブロック図

5. アプリケーションの実装

5.1. 開発環境

スマートフォン／タブレット側とサーバー側のアプリケーションを実装するための開発環境を表 5-1 に示す。本研究では前者は Android 5.0 端末を対象とする。

表 5-1 開発環境

#	項目	スマートフォン／タブレット環境	サーバー環境
1.	OS	Windows 10 Home	Windows Server 2012 R2
2.	開発ツール	Eclipse Java EE IDE for Web Developers 4.4.2	
3.		JDK 1.8.0_60	
4.		Android NDK 13.1.3345770	apache-tomcat-8.5.13
5.		Android SDK(TargetVersion 5.0.1)	UWSC
6.	ライブラリ	android-beacon-library-2.9.2	jInstagram-master_1.1.9
7.	その他	－	FTP サービス
8.			Gramblr

5.2. コーディング

(1) スマートフォン／タブレット側

主な実装クラスを表 5-2 に示す。詳細は 11.1 章を参照。

表 5-2 スマートフォン／タブレット側の主なクラス一覧

#	パッケージ					クラス	概要
1	jp	ac	keio	haruyamalabsns		MainActivity.java	アプリのメインクラス
2					common	Constants.java	定数クラス
3						DbManager.java	DB操作を行うクラス
4						FileUtility.java	ファイル操作を行う
5						Utility.java	ユーティリティクラス
6					photo	CameraFragment.java	カメラ操作を行う
7						DataManager.java	データを管理を行う
8						TransferFileInfo.java	転送情報保持クラス
9					ref	MyNoticeService.java	ビーコン電波を受信し必要に応じて既存コンテンツを表示する
10						UpdateReceiver.java	画面にメッセージを送信する

(2) サーバー側

主な実装クラスを表 5-3 に示す。スマートフォン／タブレットから FTP で受信したコンテンツ情報を Instagram にアップロードするために, 今回 Gramblr を用いる。Gramblr を操作する API は公開されていないため，アップロードする画面操作を UWSC のスクリプト（UploadInstagram.uws）に実装する。詳細は 11.1 章を参照。

表 5-3 サーバー側の主なクラス一覧

#	パッケージ					クラス	概要
1	jp	ac	keio	haruyamalabsns		Constant.java	コンスタントクラス
2						TorafuguService.java	Instagramからハッシュタグ検索でコンテンツ情報を取得する
3						TorafuguServlet.java	スマートフォン／タブレットと送受信のやり取りをするサーブレット。
4					serverApi	ApiManager.java	サーバ側APIの管理クラス
5						CommandExecutor.java	コマンドを実行する
						DbManager.java	DB操作を行うクラス
7						ServerException.java	サーバ側の内部例外クラス
8						Utility.java	ユーティリティクラス

5.3. サーバー環境構築

サーバーではスマートフォン／タブレット間で FTP 通信を行うため，FTP サーバーをインストールする。また，Instagram への投稿は Gramblr を使った画面操作で実現させるため，その環境を構築する。尚，Instagram への投稿契機はタスクスケジューラを用いて定期的にスマートフォン／タブレットからの投稿コンテンツの確認を行い，コンテンツがあれば Instagram に投稿する仕掛けを作る。研究の再現性の観点も含め，それぞれについて記載する。

(1) FTP サーバーの構築

本研究の FTP サーバー構築手順を次に示す。

① 「サーバーマネージャー」を起動し「2 役割と機能の追加」する(図 5-1)

図 5-1 サーバーマネージャーの起動画面例

② 「役割と機能の追加ウィザード」画面で「次へ(N) >」を押下して進む(図 5-2)

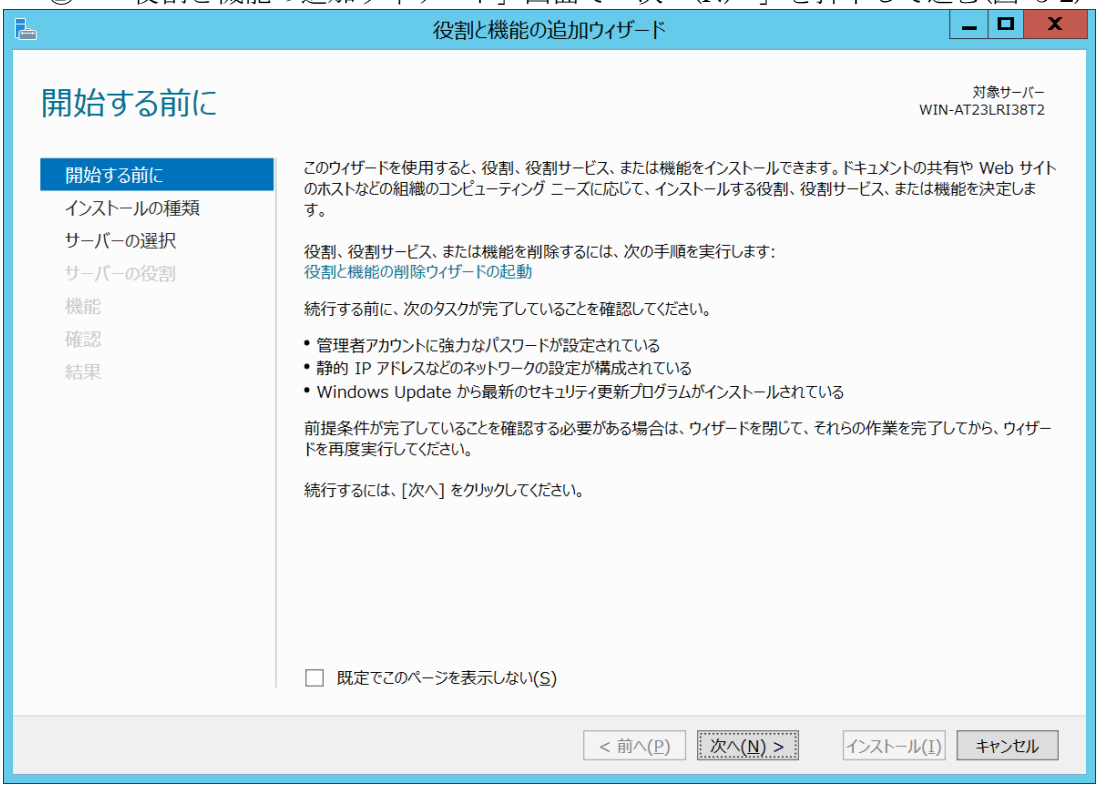


図 5-2 役割と機能のウィザード(開始する前)の画面例

③ 「役割ベースまたは機能ベースのインストール」を選択し「次へ(N) >」を押下して進む(図 5-3)

図 5-3 役割と機能のウィザード(インストールの種類)の画面例

④　一覧からインストールするサーバーを選択し「次へ(N) >」を押下して進む(図 5-4)

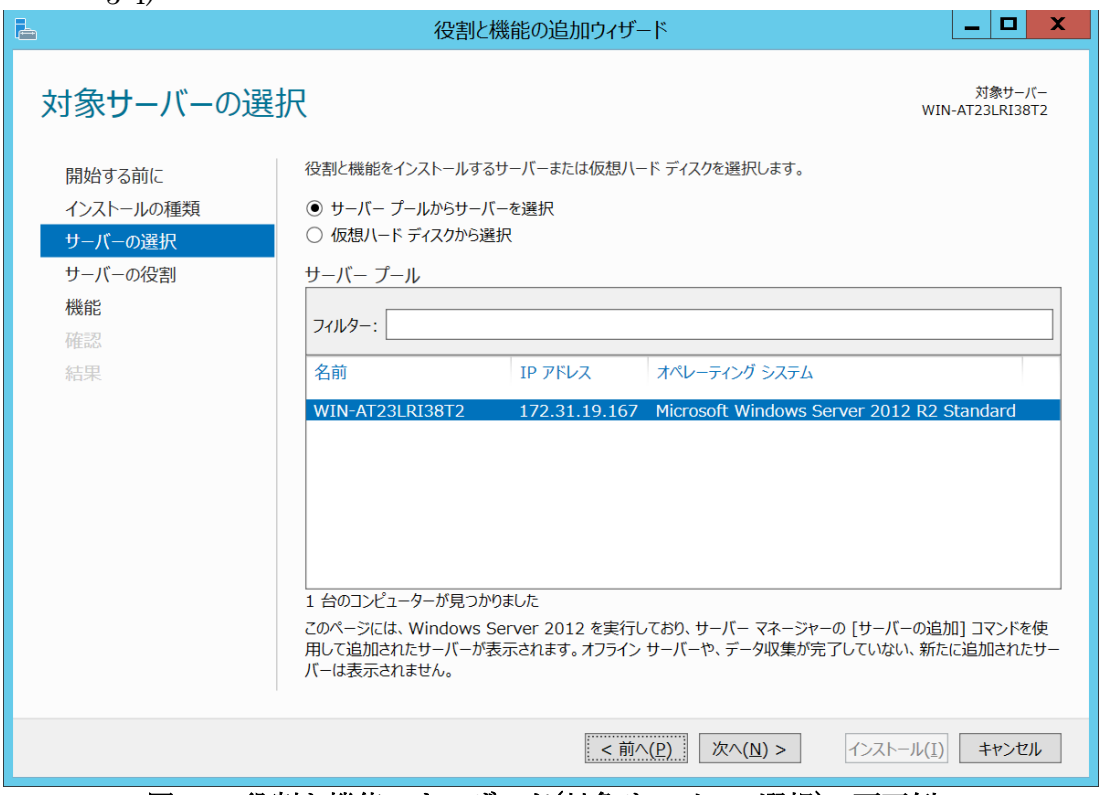


図 5-4 役割と機能のウィザード(対象サーバーの選択)の画面例

⑤　FTP サーバーをインストールするので「Web サーバー(ISS)」をチェックして機能の追加を行う。(図 5-5)

図 5-5 役割と機能のウィザード(サーバーの役割)の画面例

⑥　既に該当機能にチェックがはいっているので「次へ(N) >」を押下して進む(図5-6)

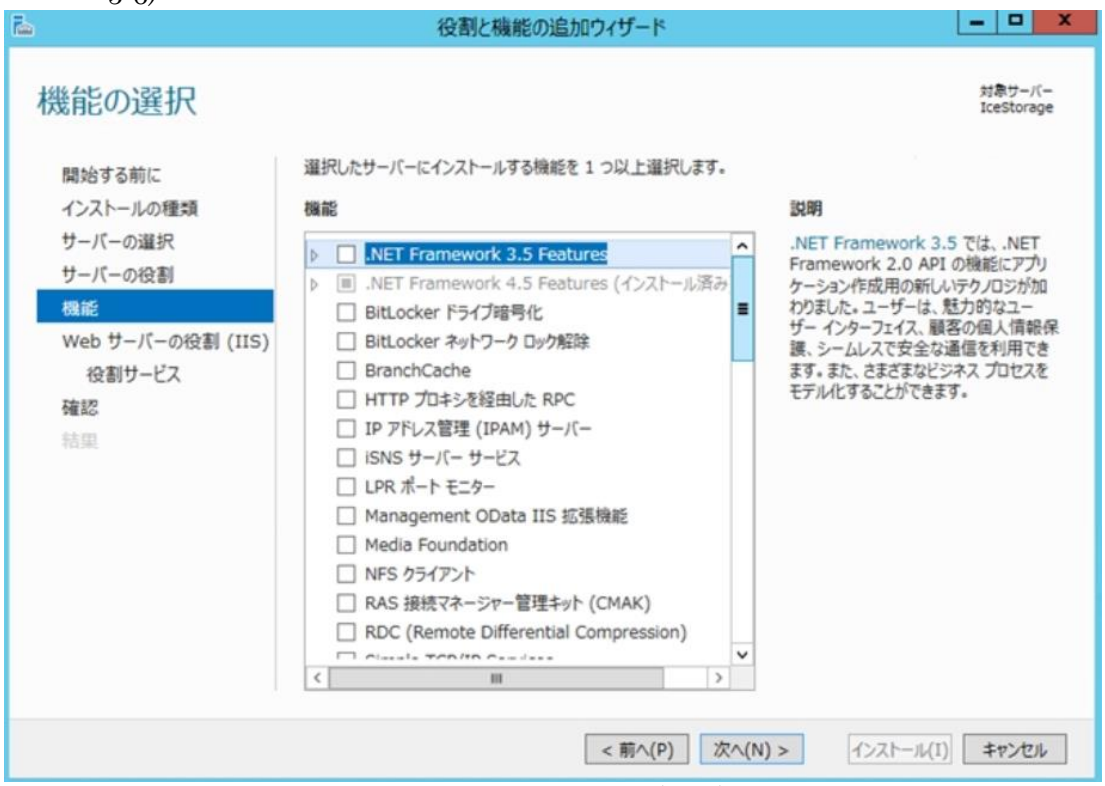


図 5-6 役割と機能のウィザード(機能)の画面例

⑦　「Web サーバーの役割(IIS)」が表示されるので「次へ(N) >」を押下して進む(図5-7)

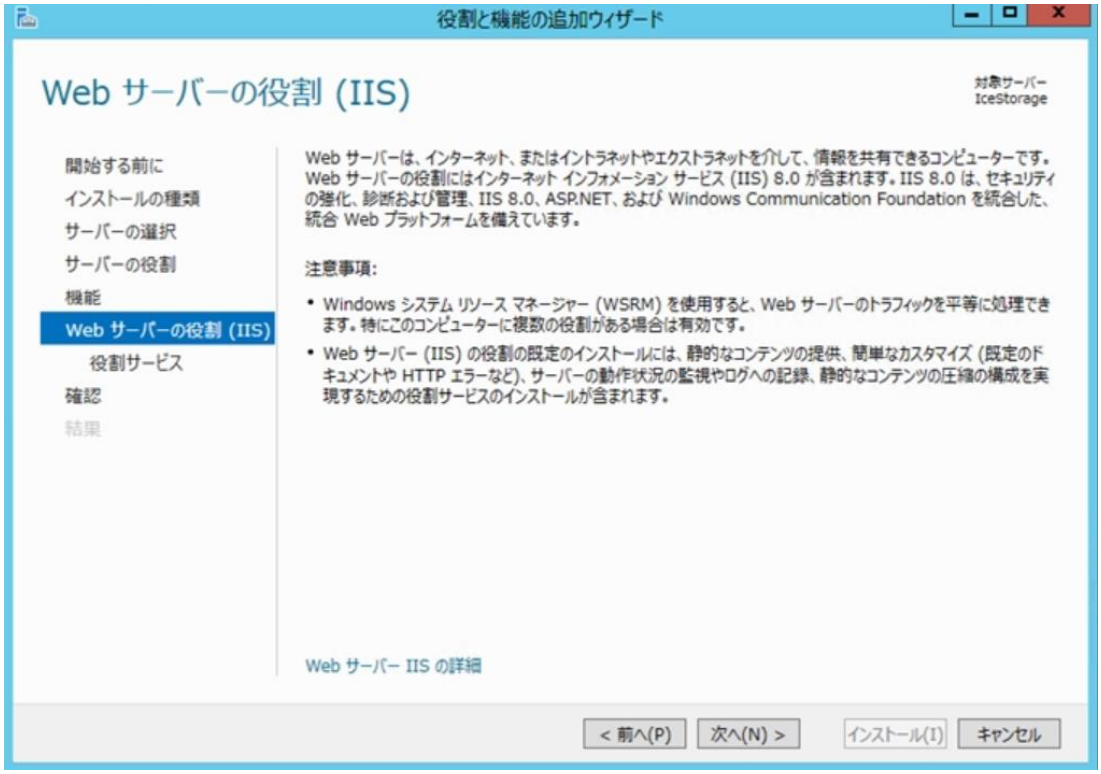


図 5-7 役割と機能のウィザード(Web サーバーの役割)の画面例

⑧　「FTP サーバー」にチェックを入れると，「FTP サービス」もチェックされるので「次へ(N) >」を押下して進む(図 5-8)

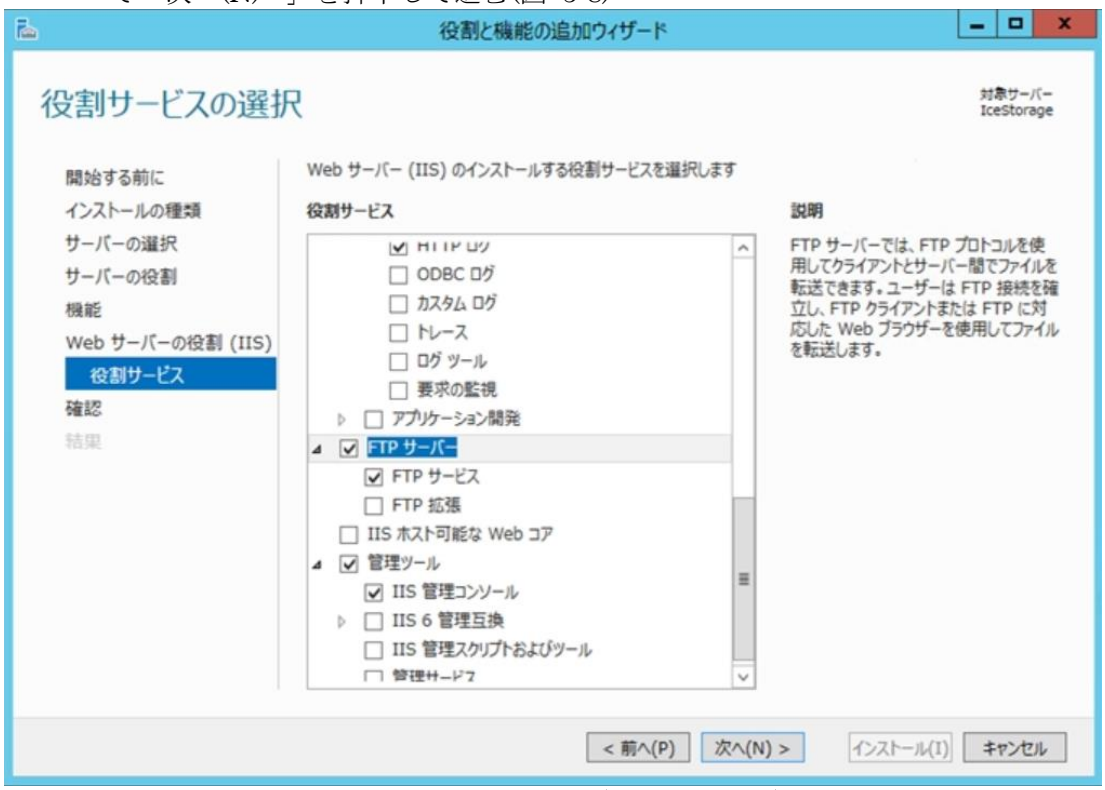


図 5-8 役割と機能のウィザード(役割サービス)の画面例

⑨　「インストール(I)」を押下し，インストールする(図 5-9)

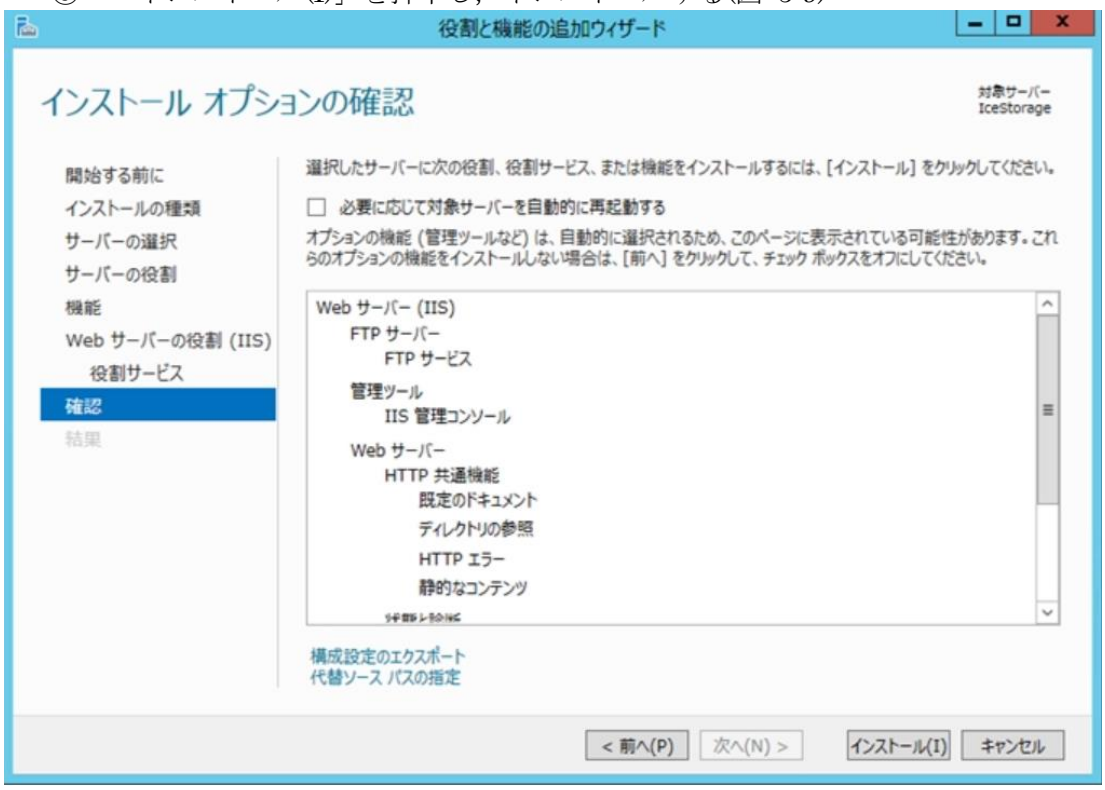


図 5-9 役割と機能のウィザード(確認)の画面例

⑩　「インターネット インフォメーション サービス(IIS) マネージャー」を起動し，FTP サイトを作成する

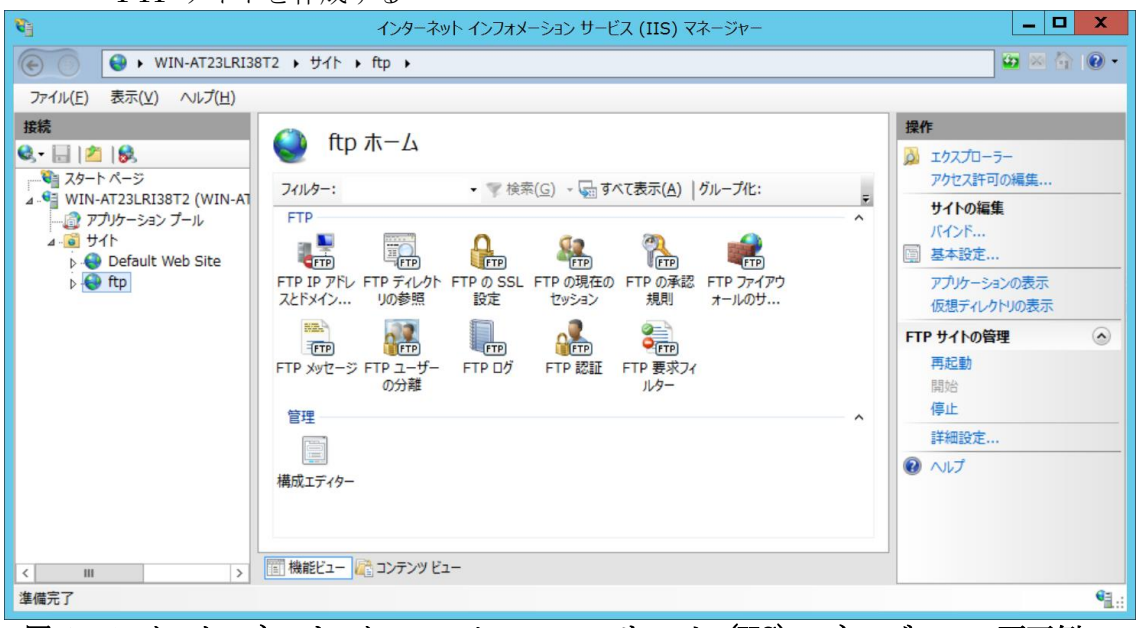


図 5-10 インターネット インフォメーション サービス(IIS) マネージャーの画面例

(2) Gramblr を用いた Instagram 投稿環境の構築

Gramblr をダウンロードし，サーバーにインストールする。Gramblr の使用には Instagram アカウントが必要になるため，今回は予め研究用に準備していたアカウントを使用し，常にログイン状態にする。

Gramblr を使用して Instagram にコンテンツを投稿する操作例を次に示す。尚，5.2 章の UWSC のスクリプト（UploadInstagram.uws）は次の操作例の自動化を実装している。

① メニューから「Upload Now!」を選択し「メディアを選択する画面」を表示し，「Drag and Drop an Image or Video, or Click Here to Start.」エリアをクリックする(図 5-11)

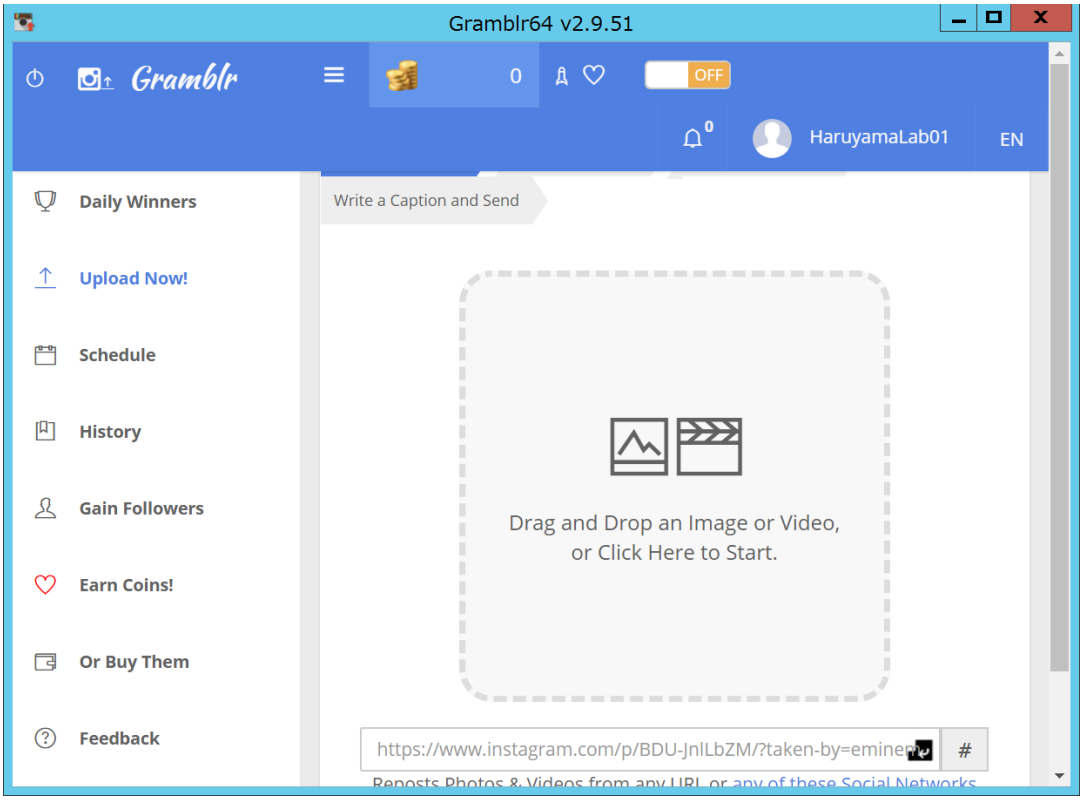


図 5-11 Gramblr のメディアを選択する画面例

② Instagram に投稿する写真ファイルを選択し，「開く(O)」を押下する(図 5-12)

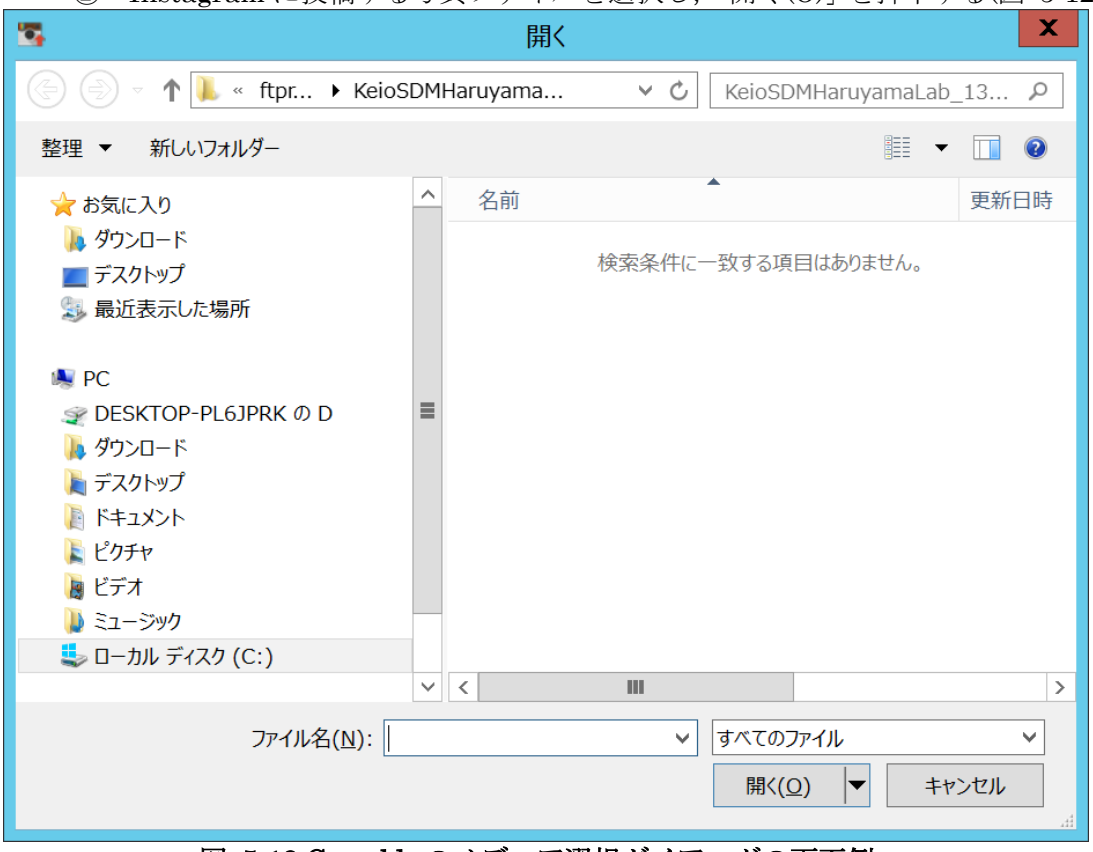


図 5-12 Gramblr のメディア選択ダイアログの画面例

③　メディアが画面に表示されるので「Save」を押下し進む(図 5-13)

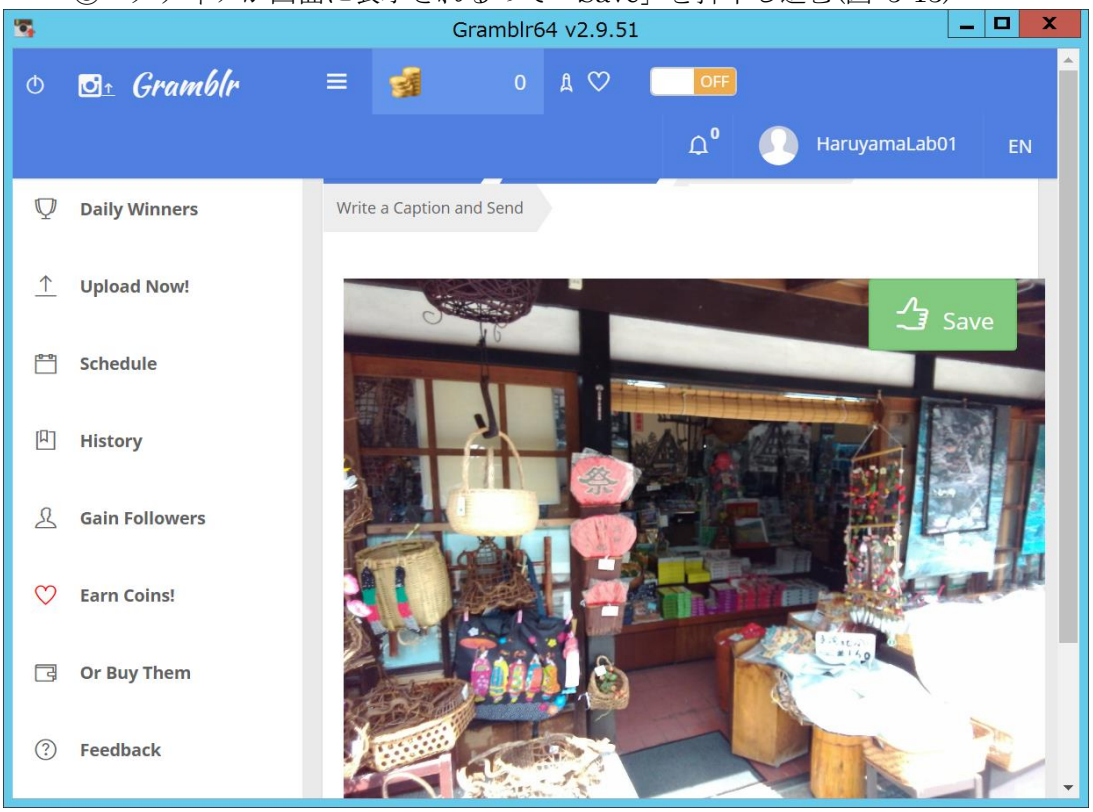


図 5-13 Gramblr のメディアサイズ編集画面例

④　フィルター機能は使用せず「Continue」を押下し進む(図 5-14)

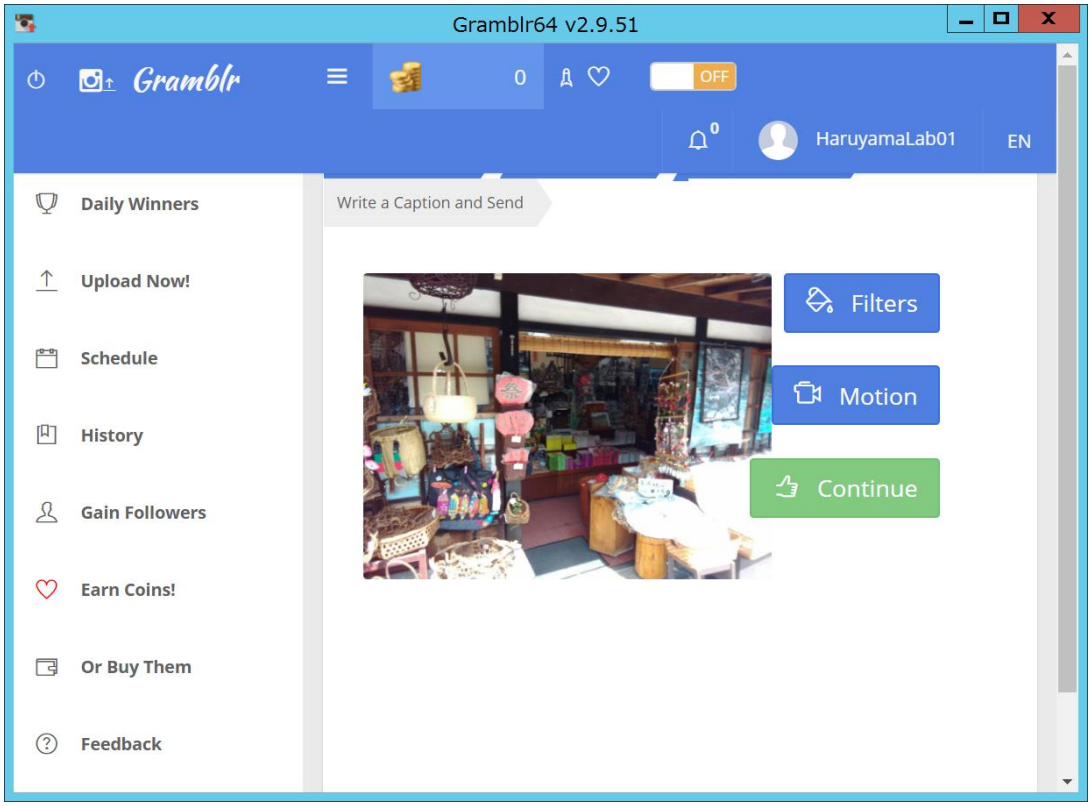


図 5-14 Gramblr のフィルター編集画面例

⑤　説明を入力する画面が表示される(図 5-15)

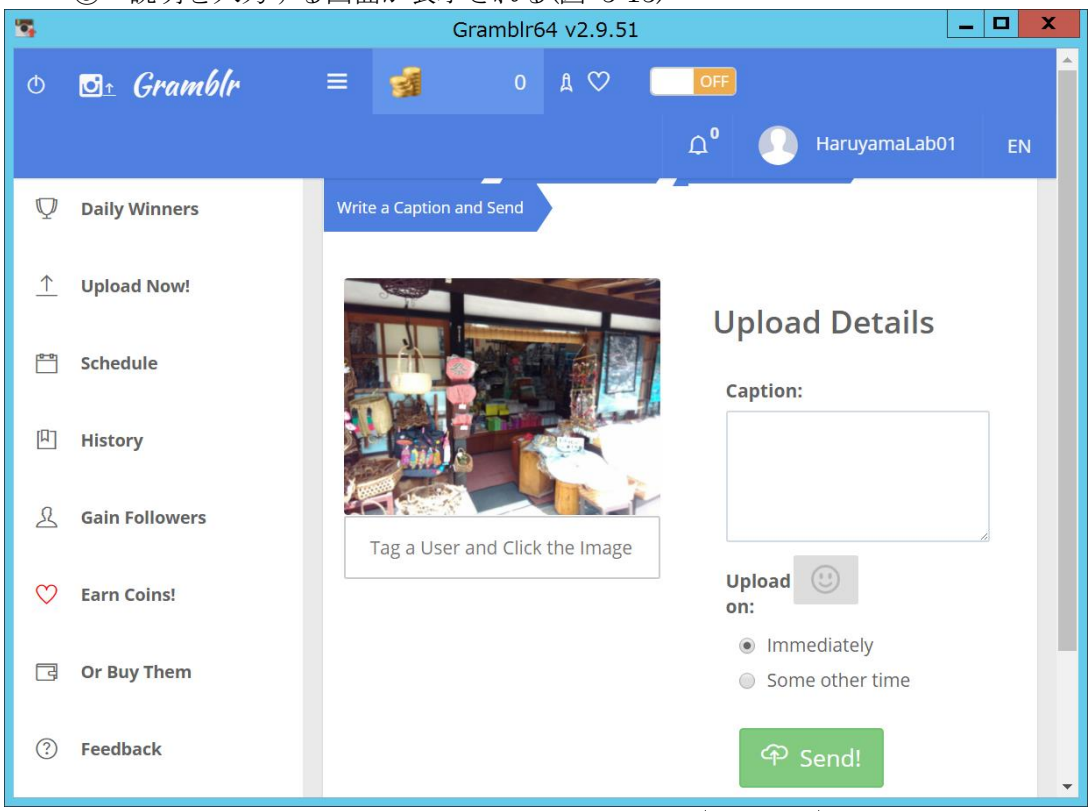


図 5-15 Gramblr の説明入力画面例(説明なし)

⑥　ハッシュタグとコメントを入力し「Send!」を押下する(図 5-16)

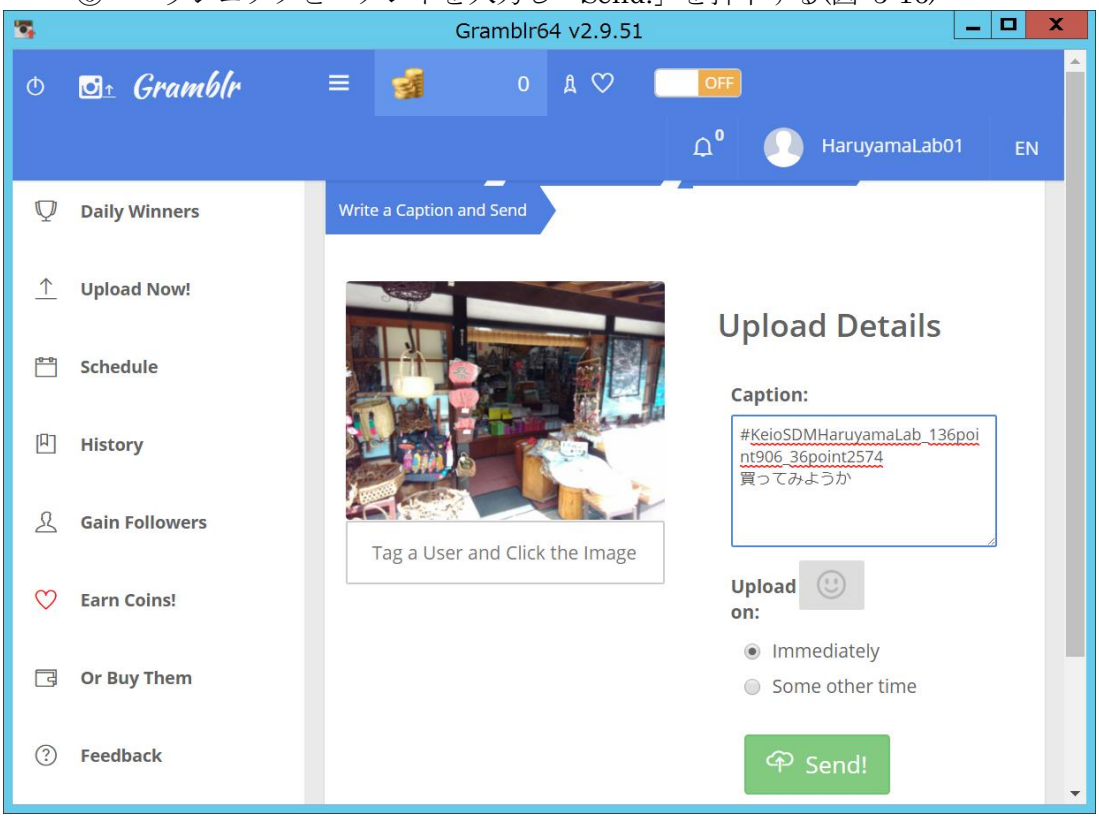


図 5-16 Gramblr の説明入力画面例(説明あり)

⑦　アップロード完了画面が表示される

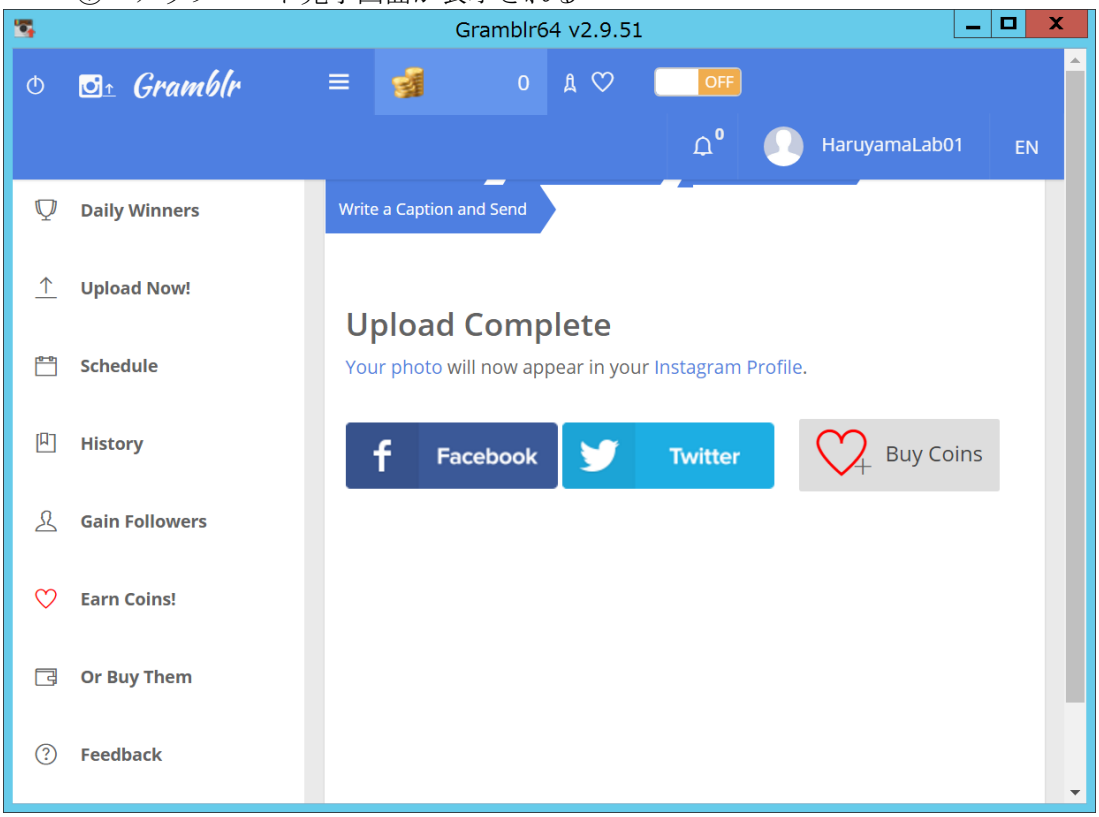


図 5-17 Gramblr のアップロード完了画面例

⑧　Instagram にアクセスすると Gramblr で投稿したコンテンツを確認することができる(図 5-18)

図 5-18 Gramblr のアップロードコンテンツを Instagram 上で確認する画面例

(3) タスクスケジューラによる定期実行環境の構築

スマートフォン／タブレットからサーバーに転送された写真ファイルが存在するかどうかの確認を表 5-3 の serverApi パッケージのクラスで行う。このチェックを定期的に行うため，serverApi パッケージのクラスを呼び出しする bat ファイルを準備し、タスクスケジューラに図 5-19 のように UploadFileTask として登録する。

もし写真ファイルが存在する場合は UploadInstagram.uws を実行し，自動で Gramblr の画面操作を行い，Instagram にアップする。

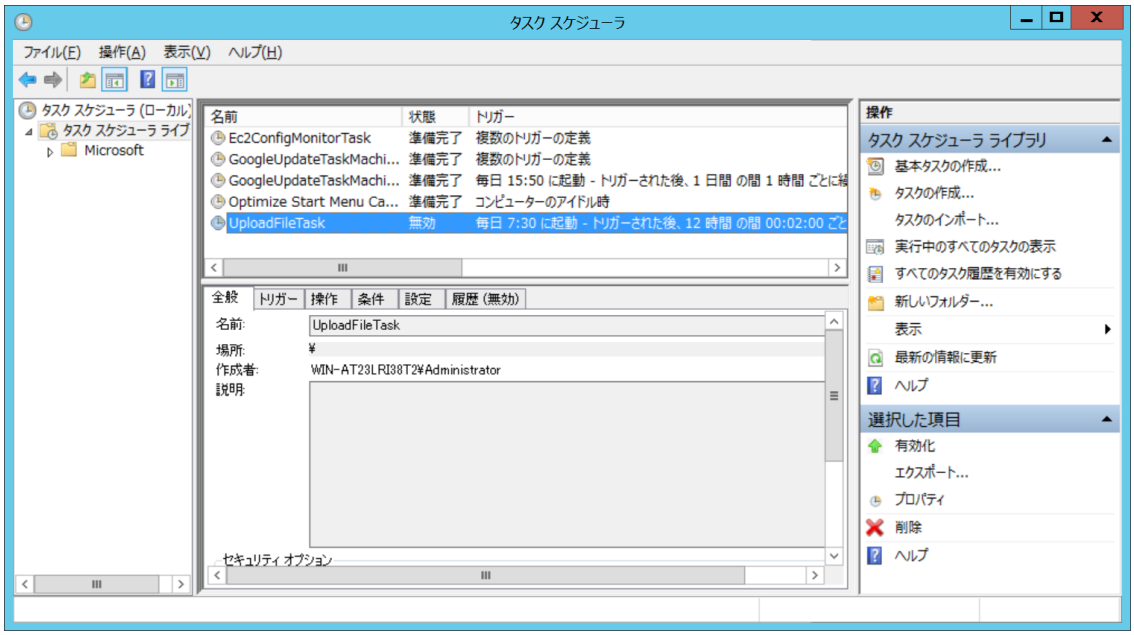


図 5-19 タスクスケジューラ ライブラリを選択した画面例

本研究ではスマートフォン／タブレット側での撮影直後に Instagram にアップするのではなく，少しタイムラグを設け，図 5-20 のように登録するタスクのトリガーを「繰り返し間隔(P)」2 分間として設定した。実際に商品として使う場合は，撮影後，速やかに SNS にアップロードするスピード感が求められるが，プロトタイプとして本システムを使用する範囲であれば，この設定でも十分検証できると判断した。

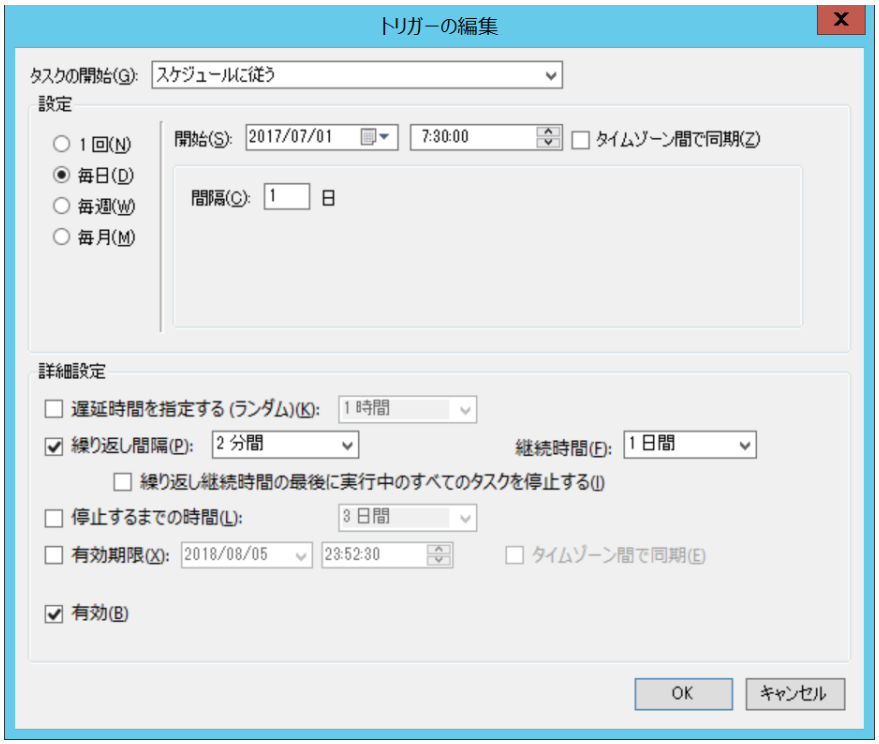


図 5-20 UploadFileTask のトリガーの編集画面例

(4) Servlet 実行環境の構築

スマートフォン／タブレットから HTTP 通信による撮影エリアの写真履歴参照リクエストを受信し，写真コンテンツをレスポンスする Servlet 環境を構築するために，Apache Tomcat をサーバ環境にインストールする。Servlet は Instagram API を使用して Instagram に投稿されている写真情報をハッシュタグ検索し，その結果をスマートフォン／タブレット側にレスポンスする。

5.4. 実装したアプリケーション

実装したアプリケーションの画面例を図 5-21 と図 5-22 に示す。Location Information を ON にしている時は，GPS から位置情報を取得するか，Beacon から電波を受信するまではカメラを起動できない。Location Information を OFF にしている時は位置情報を使用しないので，常時カメラを起動できる。また，GPS で位置情報を取得する際に，20m×20m または 100m×100m の範囲を選択することができる。(検証で使用する際は，利用者の手間を軽減させるため 20m×20m の固定にし，GPS Area ボタンは無しにした。)

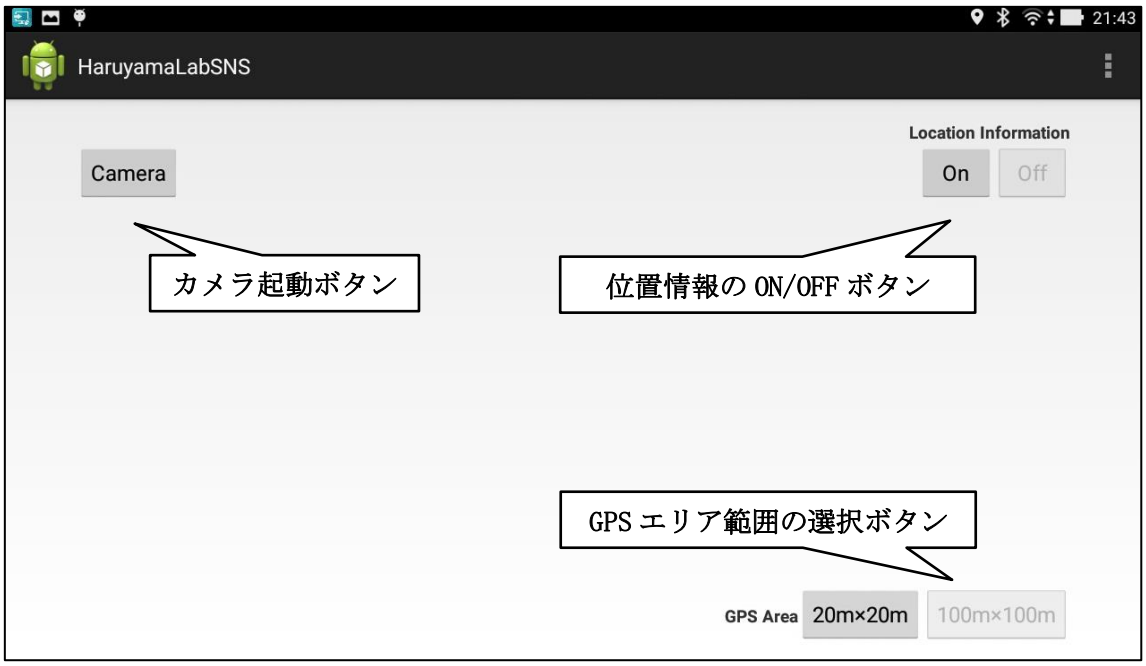


図 5-21 アプリの画面例

Location Information を ON にしている時に，位置情報を取得すると，ハッシュタグと GPS から取得した緯度，経度，Beacon 情報を図 5-22 のように画面に表示する。

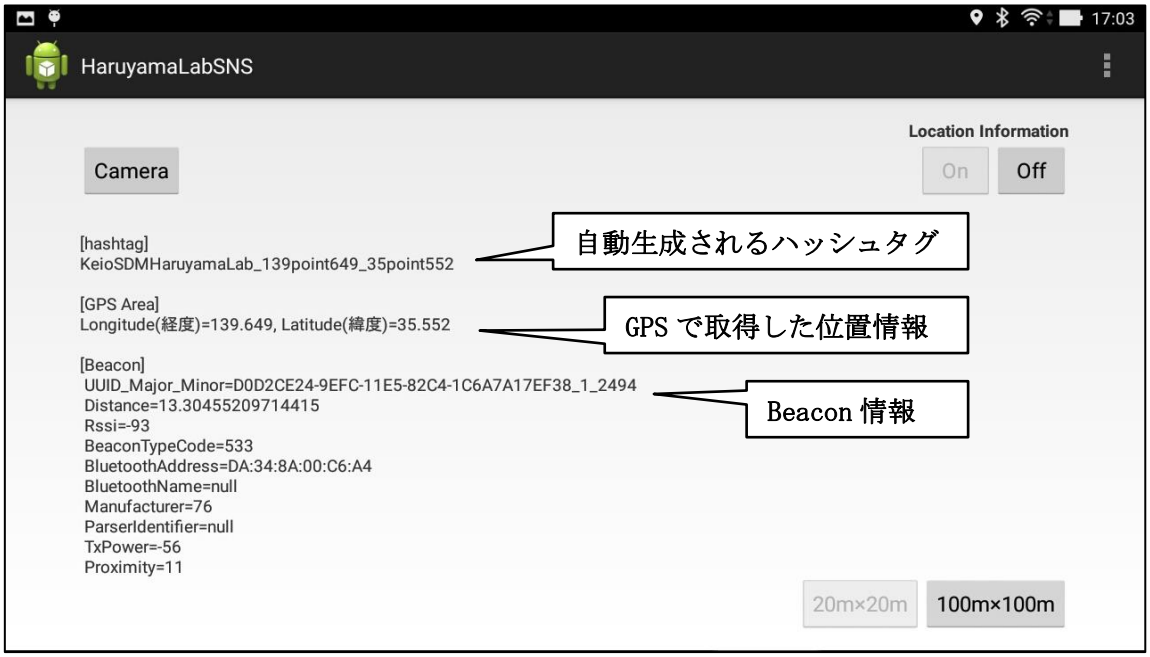


図 5-22 位置情報を取得した時の画面例

カメラを起動した時の画面例を図 5-23 に示す。

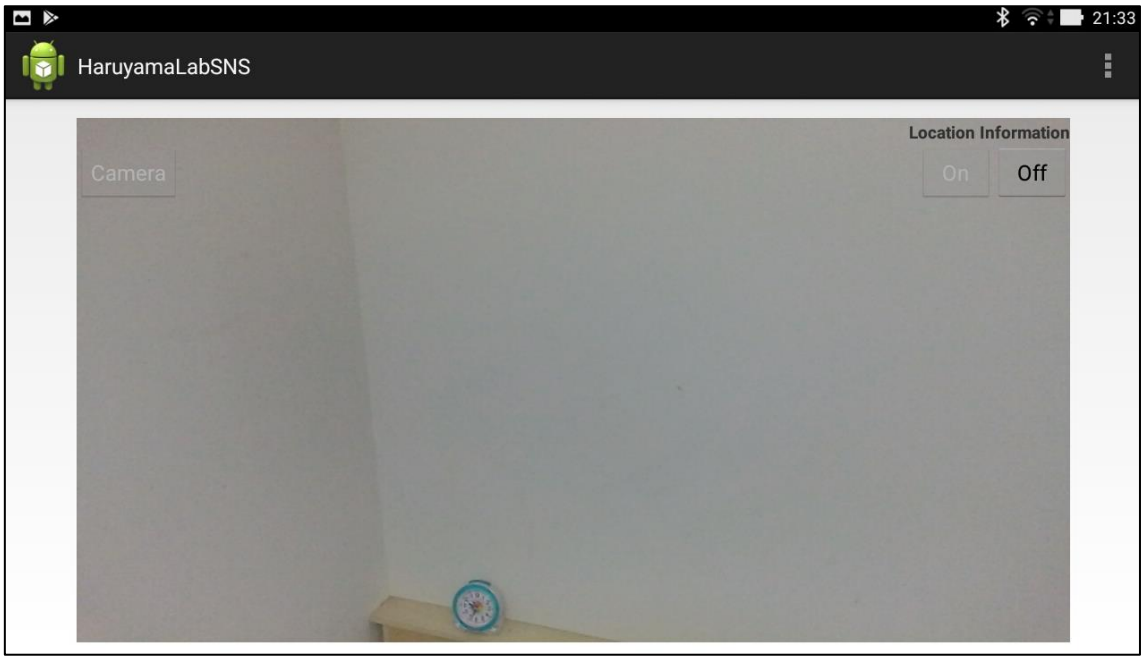


図 5-23 カメラ起動画面例

撮影後は図 5-24 のようにコメントを入力することができる。OK を押すと Instagram に写真とコメントが投稿され, Cancel を押すと Instagram の投稿を止めることができる。

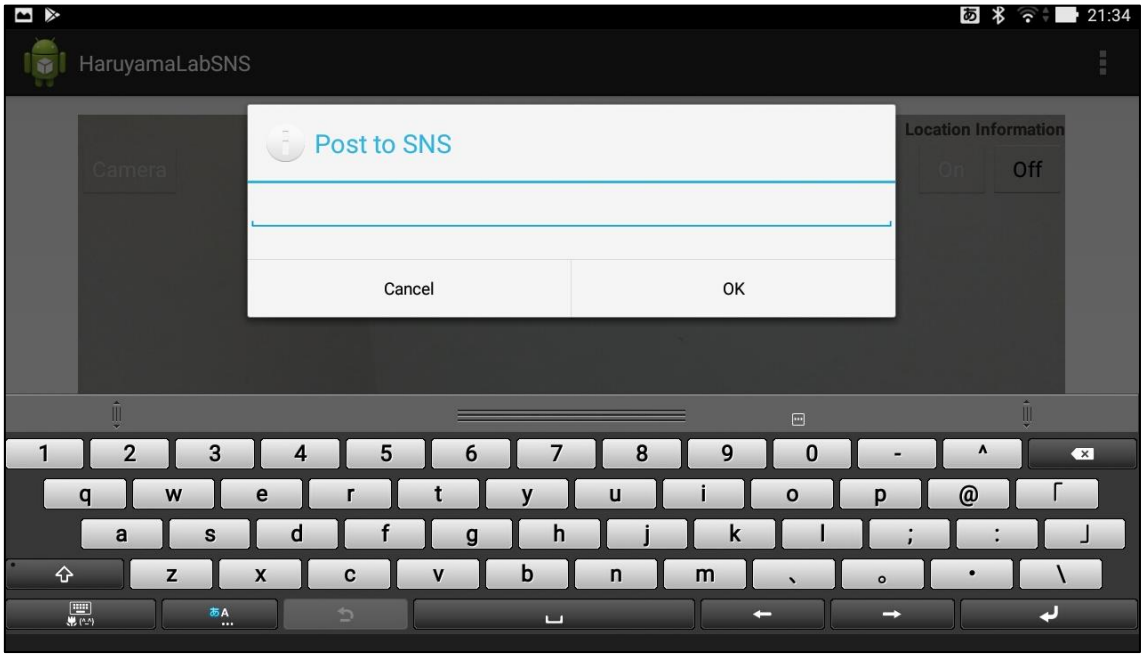


図 5-24 コメント入力画面例

位置情報を取得した状態で撮影すると，自動生成されるハッシュタグで自動的に Instagram に投稿される。図 5-25 は屋内であるが GPS で位置情報を取得できた時の撮影風景。

図 5-25 アプリを使った撮影風景

撮影後，同じ GPS エリアに撮影者が入ると自動でブラウザが起動し，その GPS エリアの写真履歴を図 5-26 のように確認することができる。

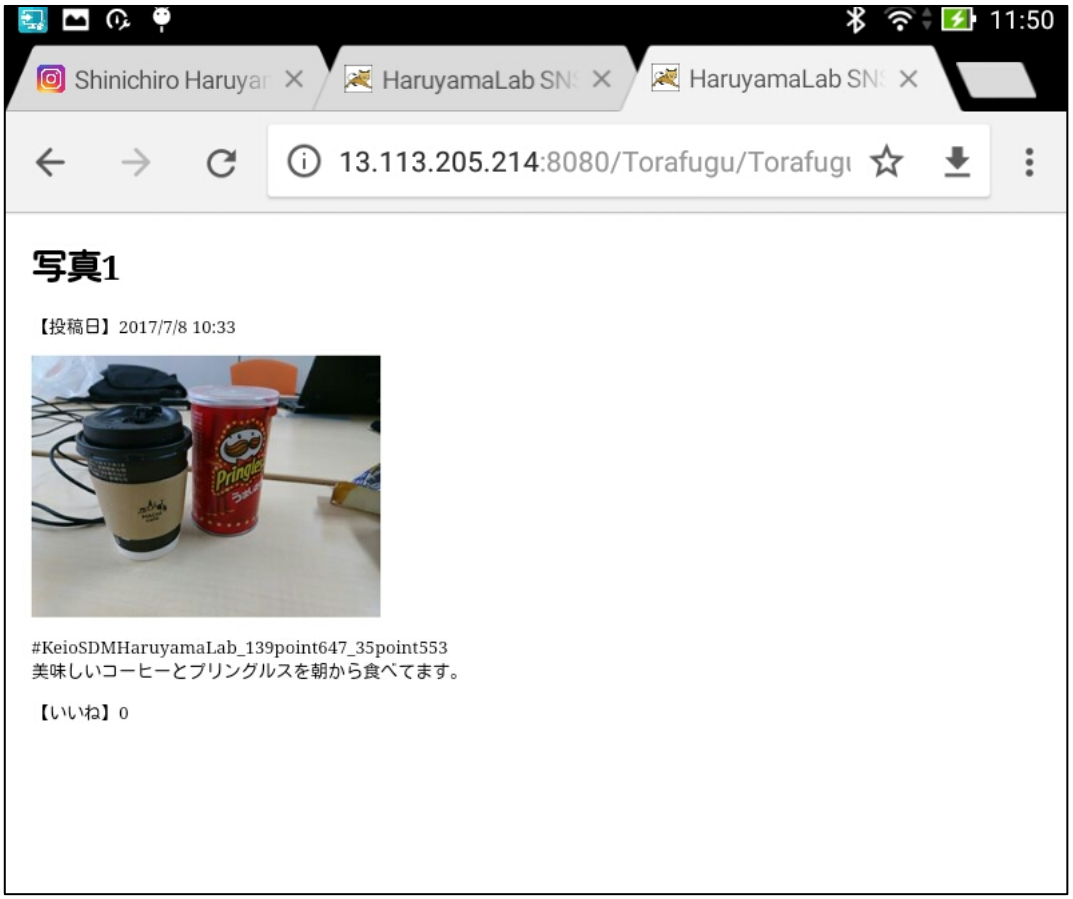


図 5-26　GPS を活用した撮影エリアで自動表示された時の例(協生館)

図 5-27 は協生館 3F の大部屋のサイネージに設置された Beacon 付近で撮影され写真が自動表示された時の例となる。

図 5-27　iBeacon を活用した撮影エリアで自動表示された時の例(協生館)

白川郷の GPS エリアマップと撮影された写真がある時に自動表示された時の例をそれぞれ図 5-28～図 5-30 に示す。

図 5-28　白川郷での GPS エリアマップ例

図 5-29　GPS エリア①で撮影済みの写真が自動表示された時の例(白川郷和田家周辺)

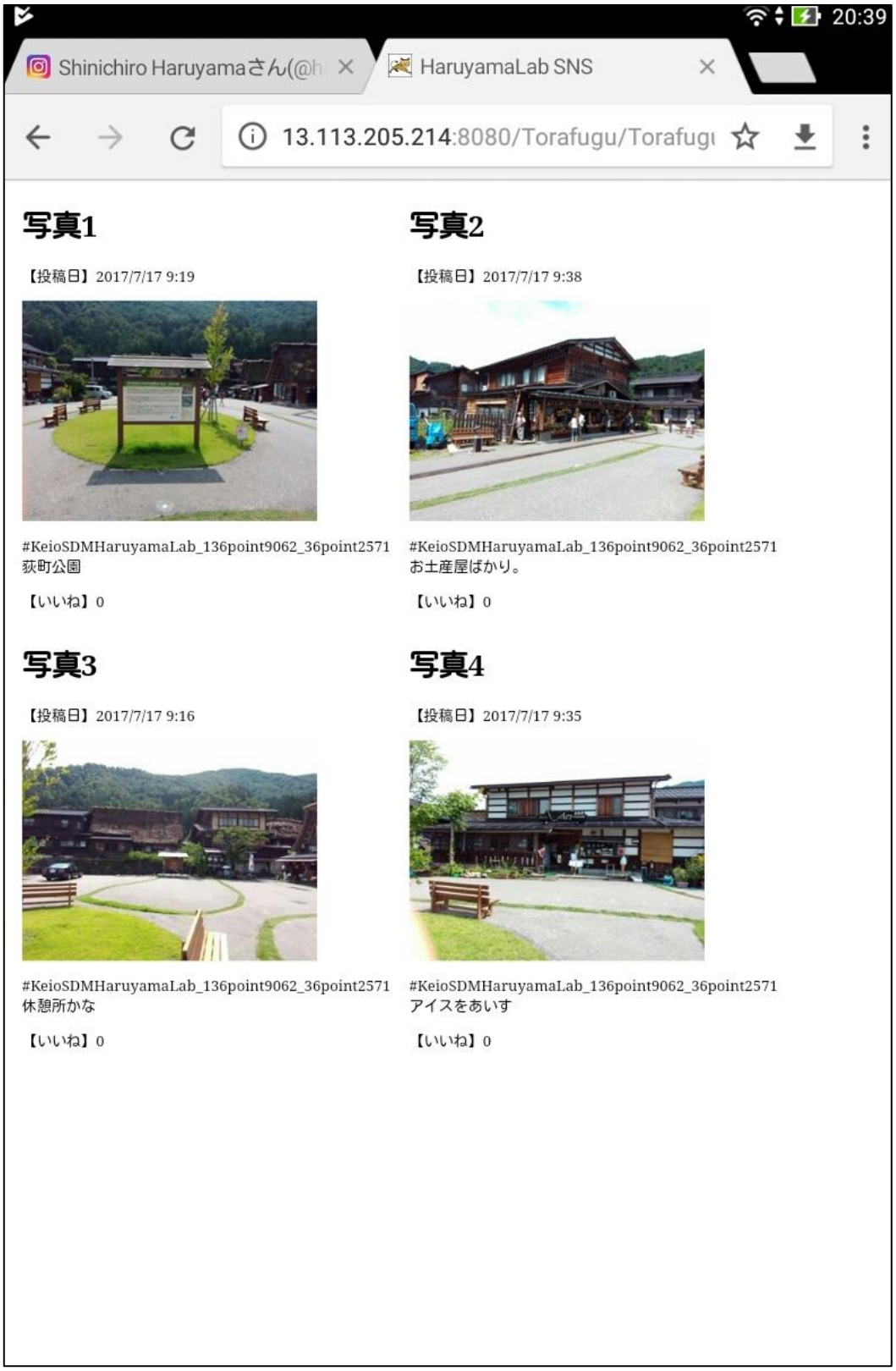


図 5-30　GPS エリア②で撮影済みの写真が自動表示された時の例(白川郷萩町公園)

6. システムの検証と妥当性確認

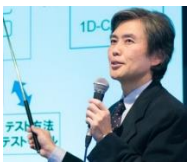
6.1. 検証

実装したアプリケーションを Android 端末にインストールし，下記要領で SDM 倫理委員会から許可をとり，主に協生館 3F と白川郷にて今回提案するシステムの確認を行った。

検証期間(予定)	7/8(土)～7/14(金)
検証の目的	既存の SNS を利用したサービスでは，位置情報を活用しているものの，写真などのコンテンツを投稿する際は利用者が位置情報を意識して入力，もしくは，端末が表示する位置スポットを選択する必要があります。例えば観光や旅行などで共有したい写真を投稿したい場合などに，その都度，位置情報を入力する手間があり，利用者負担の観点でユーザビリティの改善の余地があると考えています。 研究では既存のサービスよりも手軽にグループ内にシェアでき，その場所にいくことでグループの写真履歴を手間をかけず閲覧できるようなソリューションを提案しています。これは効率的に位置情報を活用していると考えています。今回準備しているプロトタイプのアプリではそのソリューションを簡易的に体感することができると思います。 本アプリを体感して頂き，既存の位置情報を活用したサービスと比べて，良かった点や改善点について FeedBack して頂けることを本検証の目的としています。
実施して頂きたい内容	・普段の SDM ライフを撮影しコメント付きでグループにシェアしましょう。 ・屋内の撮影対象の場所は C3S10，C3N15，CSN14 のみとなります。それぞれの部屋でサンプルコンテンツを 1 枚ずつ登録していますので参考にして下さい。投稿した写真は部屋ごとに溜まっていきます。 ・屋外は通学路でグループでシェアしたい場所を撮影して下さい。シェアしたい場所がなければ，日吉キャンパス内の任意の場所で撮影して下さい。 ・お手数ですが，検証後にアンケートを通じて Feedback をお願いします。
アプリについて	・アプリを起動して GPS による位置情報か Beacon の電波を受信するとカメラを起動できるようになります。 ・写真を撮って，コメントを入力すると自動で位置情報と紐づけして Instagram に投稿されグループ内でシェアされます。(今回の公開範囲は検証グループメンバーのみです。) ・屋内では指定の部屋で Beacon 電波を受信できる場所に行くと，電波を初回認識時に自動でブラウザが起動されてグループメンバーが撮影した画像とコメントを見ることができます。同じエリアで再度閲覧したい(ブラウザを起動したい)場合は位置情報を OFF→ON することで表示されます。 ・屋外は 20m×20m のエリアで場所と写真を紐づけます。一度そのエリアで撮影して投稿すると，メンバーがその場所に行った時に自動でブラウザが起動され，撮影画像が表示されます。 ・ブラウザが初回起動される時に Instagram の認証画面がでます。お手数ですが，下記アカウントでログイン下さい。尚，下記アカウントで Instagram にログインした際，投稿されている画像のハッシュタグの編集や削除はしないようにして下さい。 HaruyamaLab01/HaruyamaLab
アプリ使用時の前提条件	・ネットワークに接続できること ・位置情報を ON

	・Bluetooth を ON (WiFi を使用すること推奨します。)
制限事項	・一度に参照できる画像は最大 20 枚までです。 ・グループ内で 1 時間に画像を参照できる回数は最大 500 回までです。上限を超えるとアプリが正常に動かなくなります。 ・グループ内で同時に Beacon 受信エリアに入った時に正常に写真が表示されない場合があります。 ・写真が投稿できない場合は，数回程度リトライ下さい。(リトライしてもできない場合はその日の検証は終了下さい。) ・屋外の位置情報は GPS から取得していますので，実際の場所と誤差が生じる場合があります。
データの取り扱いについて	被験者は実験者より実験前に研究目的，実験方法，個人情報とデータの取り扱いの説明を受けます。実験の記録データについては，番号付けを用いた匿名化を行い，個人を特定できることはありません。また，一度頂いた同意が取り消された場合，匿名化番号を含む収集したデータを破棄し，以降の研究で使用することはありません。
アプリ不具合時の連絡先	平日は直ぐに返信できない場合がありますが，可能な限りその日のうちに連絡いたします Email:jiangchuan0001@keio.jp

iBeacon 設置ルーム

事前登録画像	C3S10 	C3S15(大部屋) 	C3S10 
共有コメント	普段の SDM ライフを撮影しコメント付きでグループにシェアしよう！		
コメント例	今日の前野先生の授業は非常に哲学的。	白坂先生の助言の元、D プロでアイデア出しに挑戦。	西村先生のシステムズエンジニアリング学で SoS の概念を理解。

検証ではまず実装したシステムが仕様通りに動くかどうかをテストした。Android7.0 では一部の機能が正常に動かない現象を確認することもあったが，開発対象にしている Android5.0 では一通り期待している機能が正常に動くことを確認することができた。結果としてシステム要求，仕様を満たしていると判断する。検証結果一覧の例を表 6-1 に示す。

表 6-1 検証結果一覧

#	識別子		試験方法概要	検証結果
	テスト	機能		
1.	TEST001	1.0	以下の情報が取得できるか確認する ・ GPS から緯度と経度 ・ Beacon の UUID	OK
2.	TEST002	2.0	アプリでカメラを起動して撮影できるかどうか	OK
3.	TEST003	3.0	コメントが入力できるかどうか	OK
4.	TEST004	4.0	投稿前に OK か Cancel の選択ができるかどうか	OK
5.	TEST005	5.0	投稿 OK を選択した後に，投稿コンテンツが SNS 上に投稿されるかどうか	OK
6.	TEST006	6.0	エリア内に投稿済みコンテンツがあるかどうか	OK
7.	TEST007	7.0	エリア内に投稿済みコンテンツがある場合，SNS 上からコンテンツを取得できるかどうか	OK
8.	TEST008	8.0	投稿コンテンツが存在するエリア内に入った時にスマートフォン／タブレットに表示されるかどうか	OK

6.2. 妥当性確認

主に 2017/7/8～7/17 の間，協生館 3F や白川郷で約 10 名の Android 端末所有者の方にプロトタイプという形で本システムを体感して頂き，妥当性確認を行った。屋内では iBeacon を，屋外では GPS を活用した本システムに対してアンケート兼インタビューを実施し，その主な結果を表 6-2　アンケート兼インタビュー結果表 6-2 に示す。

アンケート兼インタビューの結果から本システムはユーザに受け入れられる要素がある印象を強く受けた。改善すべき意見もいくつか見られるが，全体的に「使い易くて便利」「投稿する手間が省ける」「グループの投稿内容が簡単に見ることができて楽しい」などの前向きな意見が多かった。これらの結果から本システムは既存の SNS を活用した位置情報サービスにはない改善的な効果を与える可能性があり，利用者のユーズを満足させる要素があると評価する。

検証者からの改善点を参考に今後更に検討を続けることで，利用者に対して更に満足できるサービスを提供できると判断する。

表 6-2　アンケート兼インタビュー結果

#	項目	回答			
		20 代(3 人)	30 代(3 人)	40 代(2 人)	60 以上(1 人)
1.	Instagram などの SNS を使用したことがありますか？	はい(全員)			いいえ
2.	SNS に写真投稿する時に，もっと楽に投稿できると良いと思えますか？	はい(全員)			
3.	位置情報を使用したことがありますか？	はい(全員)			
4.	位置情報を使用する際にもっと楽に投稿できると良いと思えますか？	はい(全員)			
5.	本システムで良かった点があればご記入下さい。	・使い易くて便利 ・投稿する手間が省ける ・自分でグループ分けする手間が省ける ・通常の SNS より時間をかけず，多くの写真を投稿できて助かる。沢山写真を投稿したい時に本システムは非常に有効的だと思う。 ・グループの投稿内容が簡単に見ることができて楽しい。また，グループの過去の投稿履歴を閲覧できるので，ちょっとした歴史が分かるような感じがあり，面白さを感じる。 ・自分の撮影ルートに戻った時にどこでどんな写真を撮ったかを簡単に確認できて便利。 ・投稿者がいつどこにいるのかが分かり興味深い ・本システムは色々な使い方がありそう。例えば，旅行などで使うのではなく，その場所で伝えたことのお知らせや覚書にも使えそう。ペーパレスポストイット的な使いかもできそう。認知症対策にもつかえるかもしれない。			
6.	本システムで改善点があればご記入下さい。	・過去に写真が投稿されている場所に行った時に，無条件にブラウザが起動されて写真を表示するより，お知らせ通知程度にした方がよいかも。つまり，いきなり写真を表示するか，お知らせ通知をクリックすることで写真を表示されるかをユーザに選択させた方が好ましいと思う。他のアプリの操作をしている時に急に写真が表示する動きは改善の余地があると思う。 ・特定の場所でソートや検索できると面白い。 ・GPS の位置情報に誤差があると，撮影したエリアが実際の場所とずれる場合がある。位置情報の精度改善が難しいなら誤差が許容できる範囲で撮影エリアを広げた方がいいかもしれない。 ・グループを選択できると嬉しい。つまり，グループによって表示する写真の種別を分けるとより面白いと思う。例えば同じ趣味をもつ人が楽しめる専用のグループを選択できると，ユーザに対してより効果的なシステムになると考える。			

6.3. 協生館での検証証跡(Instagram 上に自動投稿された写真例)

協生館の投稿写真と場所が自動でグループ化されている写真例を図 6-1～図 6-4 に示す。

図 6-1 C3S10 の写真例

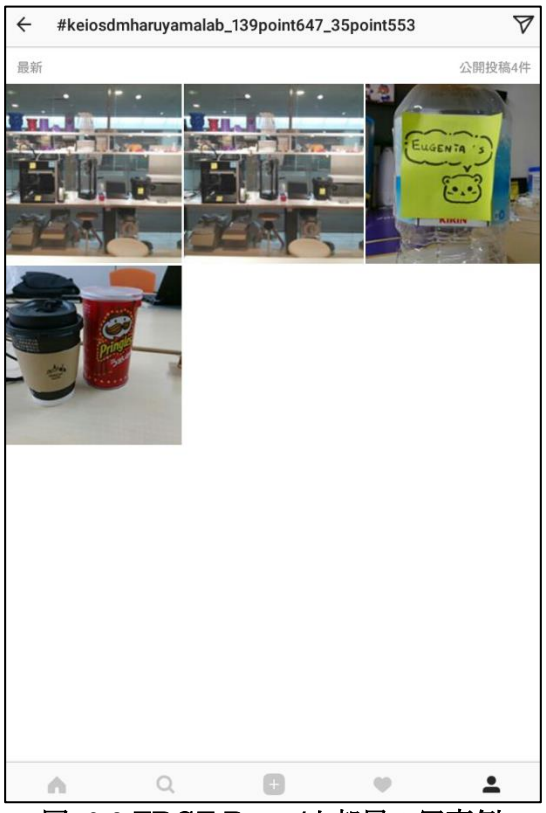


図 6-3 EDGE Room/小部屋の写真例
(緯度:35.553 経度:139.647)

図 6-2 C3N15 の写真例

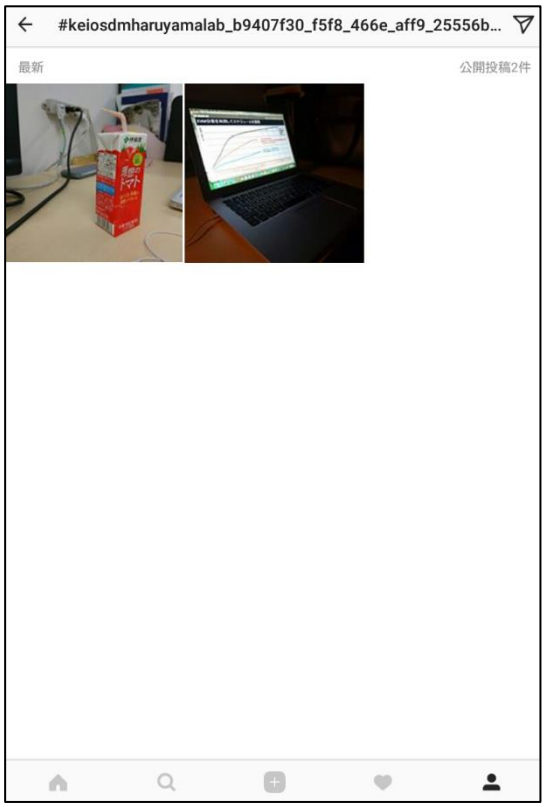


図 6-4 小部屋の写真例

6.4.　白川郷での検証証跡(Instagram 上に自動投稿された写真例)

GPS を活用した撮影エリアのマップを白川郷と五箇山でそれぞれ図 6-5 と図 6-6 に示す。

図 6-5 白川郷の撮影エリアマップ

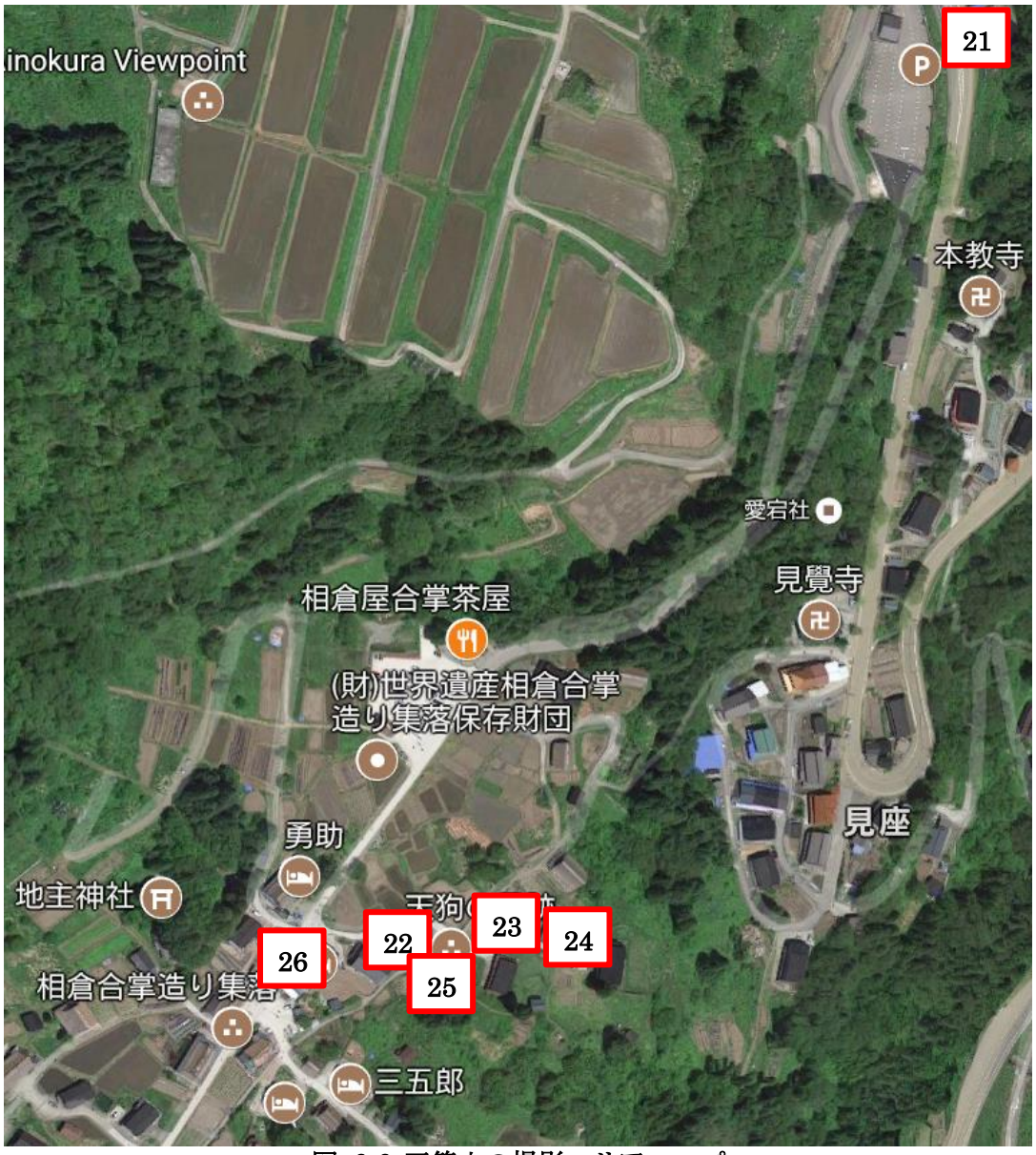


図 6-6 五箇山の撮影エリアマップ

白川郷と五箇山での投稿写真と場所が自動でグループ化されている写真例を図 6-7～図 6-32 に示す。

図 6-7 撮影エリア 1 の写真例
(緯度:36.2605 経度:136.9073)

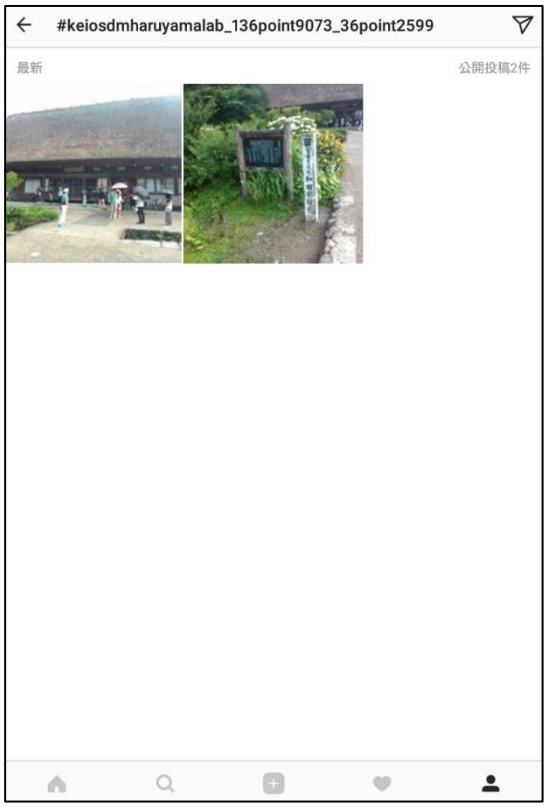


図 6-9 撮影エリア 3 の写真例
(緯度:36.2599 経度:136.9073)

図 6-8 撮影エリア 2 の写真例
(緯度:36.26 経度:136.9079)

図 6-10 撮影エリア 4 の写真例
(緯度:36.2598 経度:136.907)

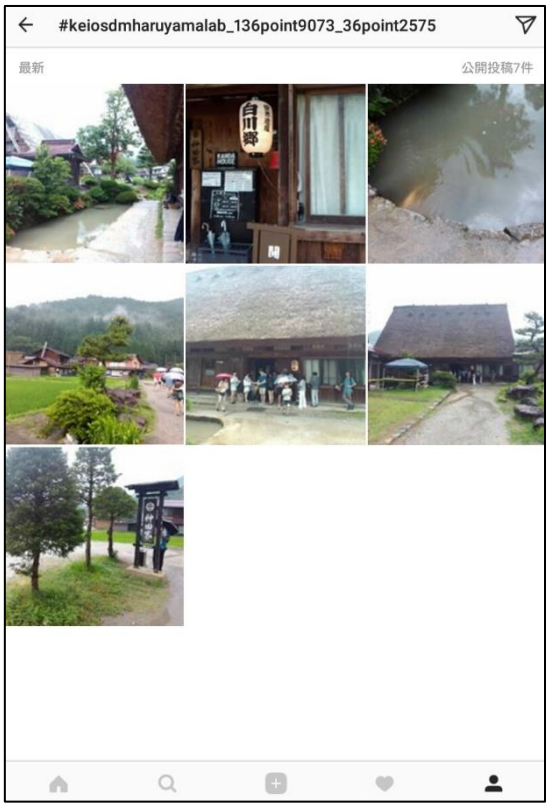


図 6-11 撮影エリア 5 の写真例
(緯度:36.2575 経度:136.9073)

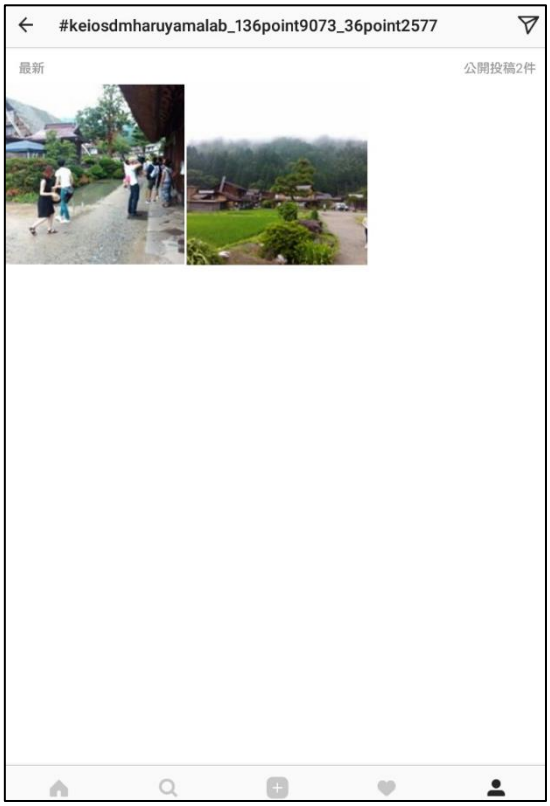


図 6-12 撮影エリア 6 の写真例
(緯度:36.2577 経度:136.9073)

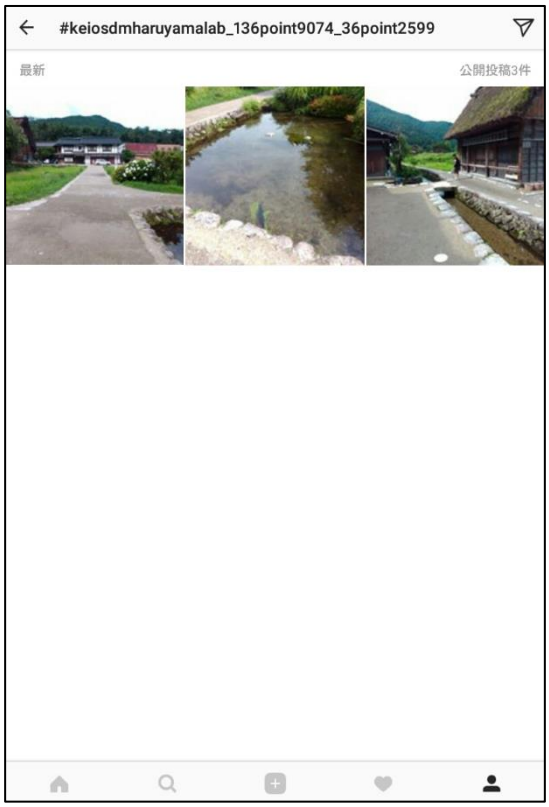


図 6-13 撮影エリア 7 の写真例
(緯度:36.2599 経度:136.9074)

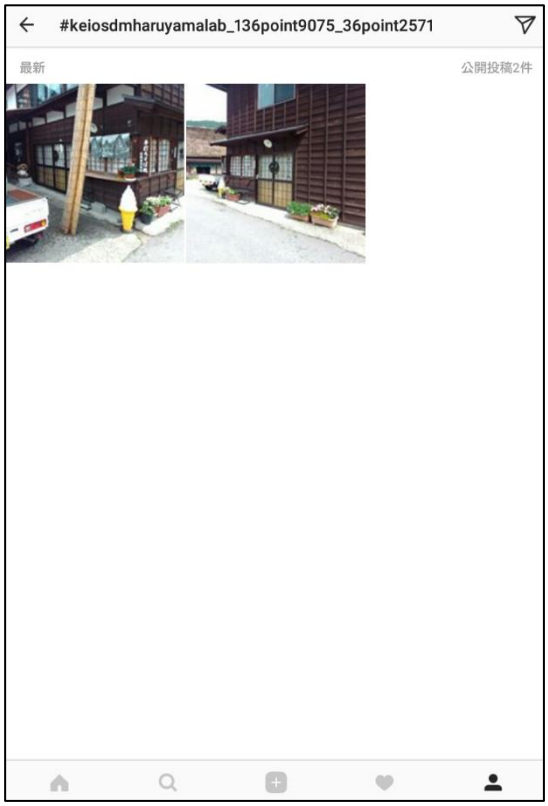


図 6-14 撮影エリア 8 の写真例
(緯度:36.2571 経度:136.9075)

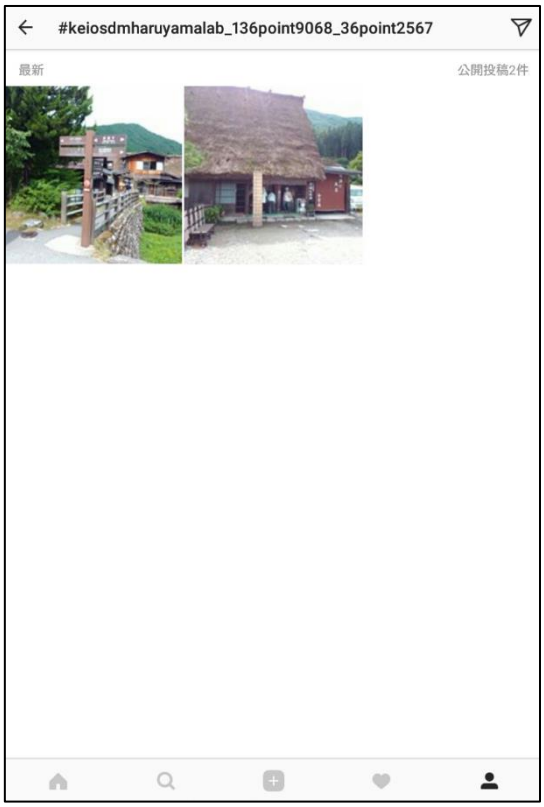


図 6-15 撮影エリア 9 の写真例
(緯度:36.2567 経度:136.9068)

図 6-17 撮影エリア 11 の写真例
(緯度:36.2557 経度:136.9063)

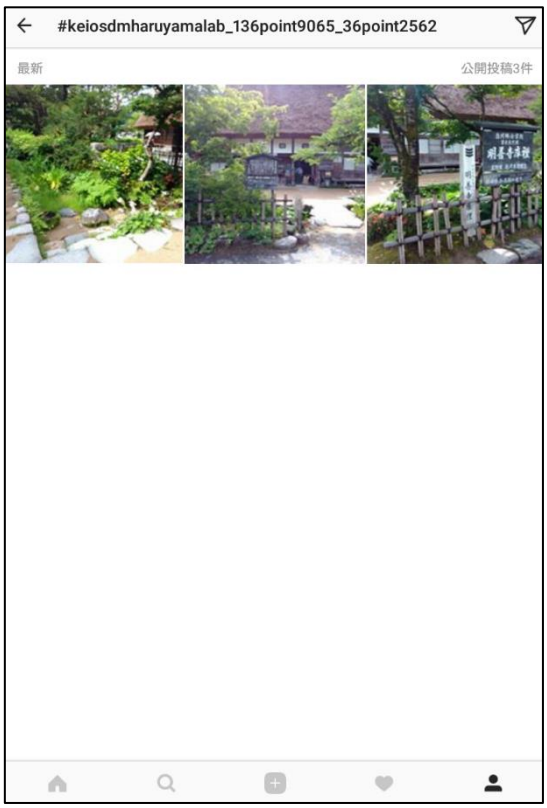


図 6-16 撮影エリア 10 の写真例
(緯度:36.2562 経度:136.9065)

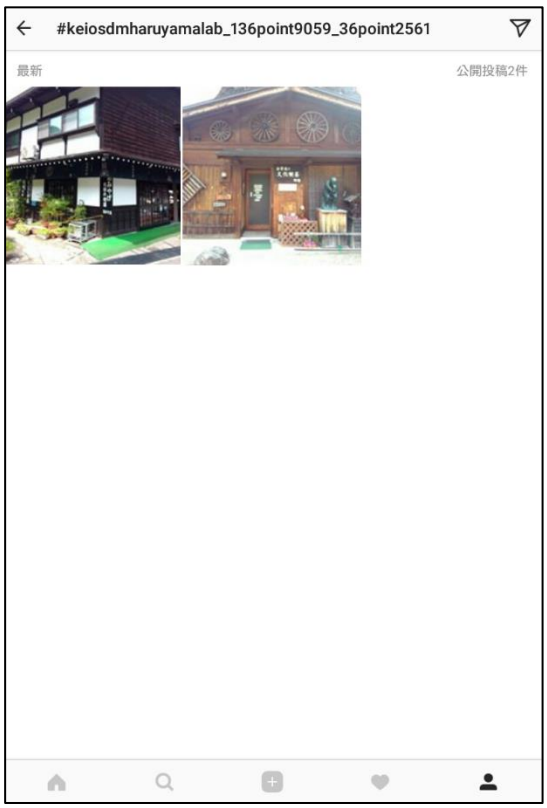


図 6-18 撮影エリア 12 の写真例
(緯度:36.2561 経度:136.9059)

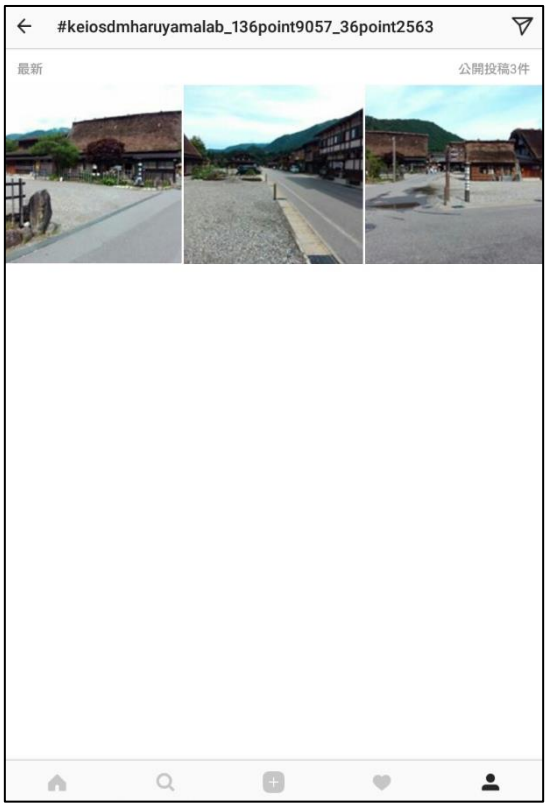


図 6-19 撮影エリア 13 の写真例
(緯度:36.2563 経度:136.9057)

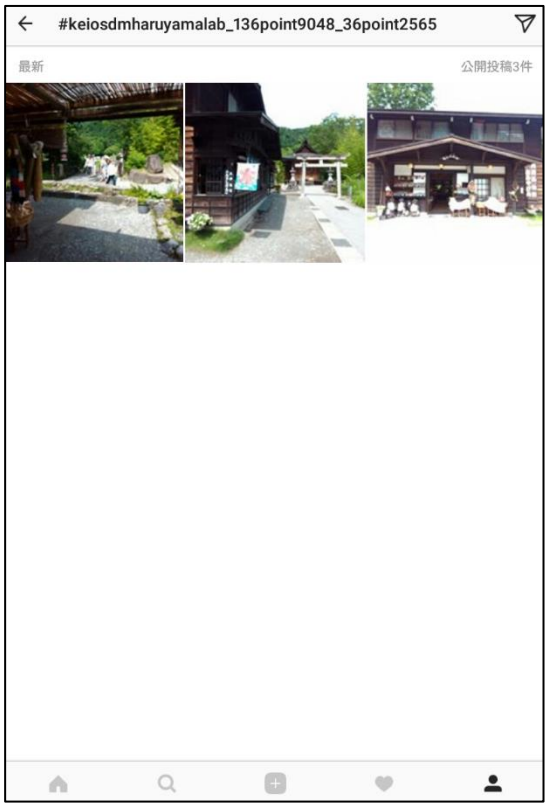


図 6-21 撮影エリア 15 の写真例
(緯度:36.2565 経度:136.9048)

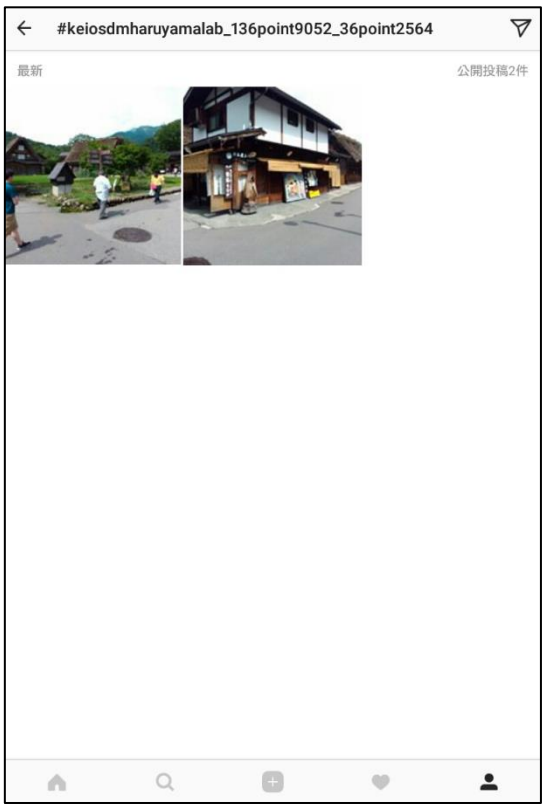


図 6-20 撮影エリア 14 の写真例
(緯度:36.2564 経度:136.9052)

図 6-22 撮影エリア 16 の写真例
(緯度:36.2571 経度:136.9062)

図 6-23 撮影エリア 17 の写真例
(緯度:36.2574 経度:136.906)

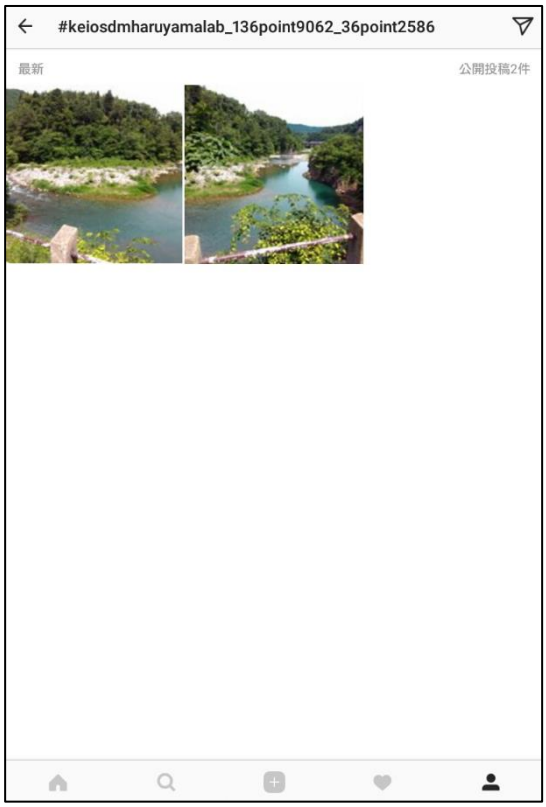


図 6-25 撮影エリア 19 の写真例
(緯度:36.2586 経度:136.9062)

図 6-24 撮影エリア 18 の写真例
(緯度:36.2579 経度:136.9057)

図 6-26 撮影エリア 20 の写真例
(緯度:36.2608 経度:136.9068)

図 6-27 撮影エリア 21 の写真例
(緯度:36.4302 経度:136.9395)

図 6-29 撮影エリア 23 の写真例
(緯度:36.4265 経度:136.9371)

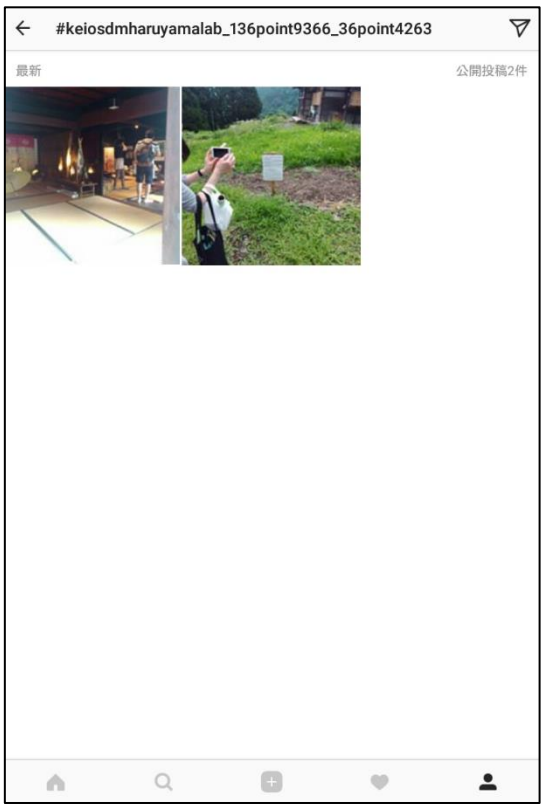


図 6-28 撮影エリア 22 の写真例
(緯度:36.4263 経度:136.9366)

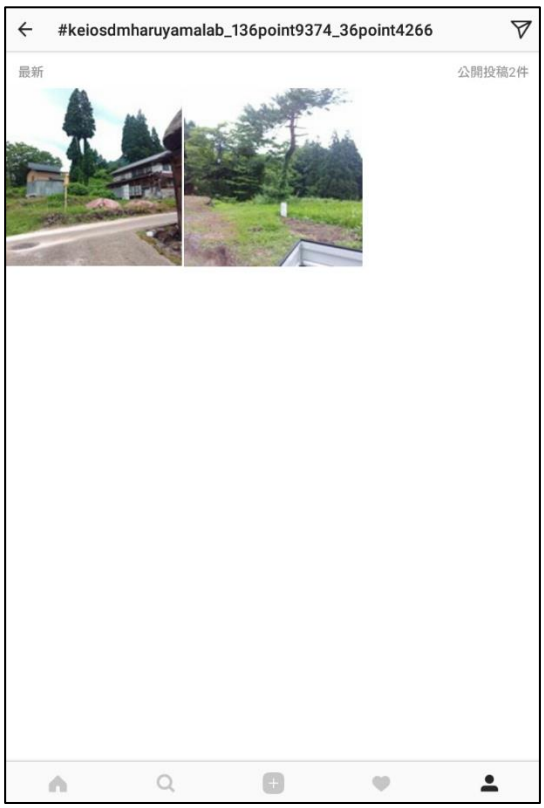


図 6-30 撮影エリア 24 の写真例
(緯度:36.4266 経度:136.9374)

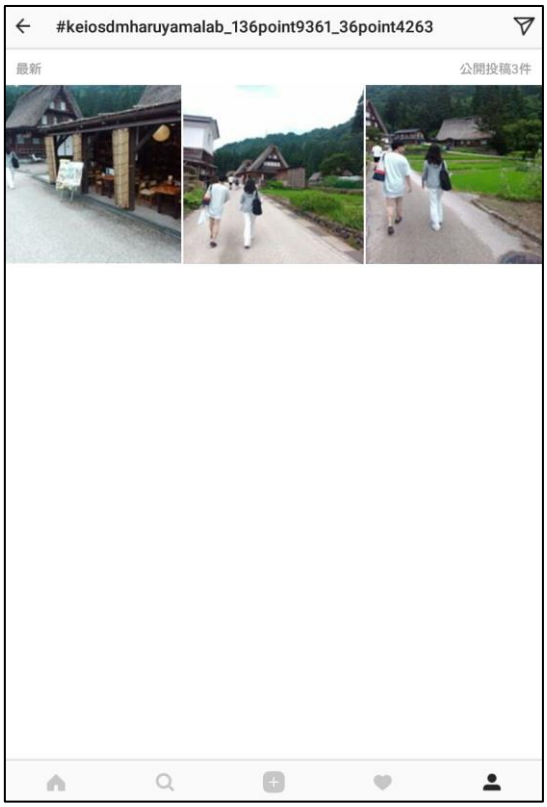


図 6-31 撮影エリア 25 の写真例
(緯度:36.4263 経度:136.9361)

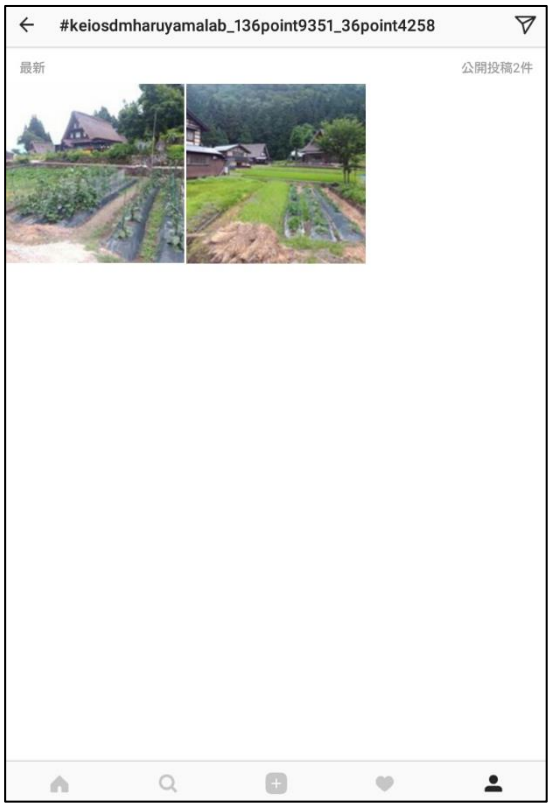


図 6-32 撮影エリア 26 の写真例
(緯度:36.4258 経度:136.93

7. 今後の課題

6.2 章の結果より，本研究で提案するシステムはユーザに受け入れられる要素があると考える。一方で改善すべき点もあり，今後検討を続けることで，利用者に対して更に満足できるサービスを提供できると考える。

また課題の一つとして，許可されたグループ内で本音ベースの会話ができるため，ネガティブな発言も含まれる場合がある。商品などを提供するコンテンツに対して，このような発言が増えると，商品の売り上げにマイナスの影響を与える可能性があるため，この事象に対する対策を検討していく必要がある。その他，今回ハッシュタグを GPS であれば緯度と経度を，Beacon では UUID を用いて作成しているが，位置スポットが分かるハッシュタグで登録するやり方も今後の課題として検討の余地があると考える。

今回，位置情報を活用したサービスの利便性を向上させることを目的としているが，それによって社会にどのような価値を提供できるのかを検討していくことが非常に大切だと考える。副査の谷口尚子准教授からも助言を頂き，本ソリューションは幅広く利用できる可能性があるとのご意見を頂きました。任意の場所に行くことで，コンテンツを容易に確認できるアプローチは，災害対策のお知らせや，警察の遺留品の捜査等にも活用できる要素があり，視点を変えることで，社会に大きな価値をもたらすソリューションに繋がりうるとの印象を持つことができた。そのようなソリューションを検討し形にしていくことも今後の課題である。

8. 結論

本研究では 1.2 章で記載している通り，既存のサービスよりユーザビリティを向上できるような提案を行うことが本研究の目的である。実際に提案するシステムを体感して頂いた方からは，「使い易くて便利」「投稿する手間が省ける」「グループの投稿内容が簡単に見ることができて楽しい」などの前向きの意見が多数あることから，ユーザビリティの改善に繋がっている印象を受けた。また，自動で位置と写真を紐づけして投稿できるだけでなく，グループメンバーの投稿スポットに行くだけで，その投稿内容を容易に確認できるアプローチは，利用者間の楽しさを助長するような要素もあると考えられる。今回挙げた今後の課題を改善することで，更に利用者の満足度を向上させるサービスを提供できる可能性があると考察する。

9. 謝辞

本研究を行うにあたりご指導・ご鞭撻を頂きました指導教員の春山真一郎教授，副査の谷口尚子准教授に心より感謝申し上げます。また，日常のミーティングを通じて多くのご意見・示唆を頂いた春山研究室の皆さまや，授業や研究を通じて助言をして頂きました関連者の皆さまに深く感謝いたします。

10. 参考文献

(1) 位置情報ビジネス報告書 2016(株式会社インプレス発行)
主に「2.1.既存のサービス」と「2.2.位置情報を活用した SNS の動向」で参考。

(2) Wikipedia:緯度・経度の調べ方
緯度・経度を用いて投稿エリアの範囲を計算する際に下記を参考。
URL
https://ja.wikipedia.org/wiki/Wikipedia:%E7%B7%AF%E5%BA%A6%E3%83%BB%E7%B5%8C%E5%BA%A6%E3%81%AE%E8%AA%BF%E3%81%B9%E6%96%B9

(3) システム×デザイン思考で世界を変える 慶應 SDM「イノベーションのつくり方」(2014)
編著者:前野隆司

(4) システムズモデリング言語 SysML(2012)
著者:Sanford Friedenthal, Alan Moore, Rick Stiner
監訳者:西村秀和

(5) 最新システムエンジニアリング情報館
http://se.rdy.jp/

(6) INCOSE SYSTEMS ENGINEERING HANDBOOK
A GUIDE FOR SYSTEM LIFE CYCLE PROCESSES AND ACTIVITIES (2015)
著者:INCOSE

(7) 本文イメージ画像の参照先 URL(2017/08/11 時点)
・Nurgle 簡単！iPhone でオシャレに写真を撮るおすすめのカメラアプリ 9 選
http://nurgle77.com/239.html
図 2-4 で参照

・BABA BARFANI EXPORTS Beacon Bluetooth Device
http://www.bababarfaniexports.com/beacon-bluetooth-device.htm
図 2-5 で参照

・京都新聞 弾丸４発，容疑男の両脚貫通　京都，被弾後も刃物捨てず
http://www.kyoto-np.co.jp/politics/article/20161130000064
図 2-6 で参照

・神戸新聞 NEWS 都賀川増水事故
http://komma.jp/archives/08togagawa/0001372826.shtml
図 2-7 で参照

・くまもり News なんと兵庫県姫路市にクマ，くまもり本部が急行
http://kumamori.org/news/category/%E3%81%8F%E3%81%BE%E3%82%82%E3%82%8Anews/38024/
図 2-8 で参照

・Yuichiro Suzuki Reports ポストイットを綺麗にデジタル化するアプリ Post-it Plus が素晴らしかった件
http://sozoen.com/yuichiro/post-it-app
図 2-9 で参照

11. 付録

11.1. ソースコード

11.1.1. スマートフォン／タブレット

（1） HaruyamaLabSNS プロジェクト

プロジェクト構成例

```
root フォルダ
|    AndroidManifest.xml
|    ic_launcher-web.png
|    proguard-project.txt
|    project.properties
|
├──assets
|        beacon.properties
|
├──gen
|    └──jp
|        └──ac
|            └──keio
|                └──haruyamalabsns
|                        BuildConfig.java
|                        R.java
|
├──libs
|        altbeacon.jar
|        android-support-v4.jar
|        commons-net-3.5.jar
|        postgresql-9.4.1212.jre6.jar
|
├──res
|    ├──drawable-hdpi
|    |        ic_launcher.png
|    |
|    ├──drawable-ldpi
|    ├──drawable-mdpi
|    |        ic_launcher.png
|    |
|    ├──drawable-xhdpi
|    |        ic_launcher.png
|    |
|    ├──drawable-xxhdpi
|    |        ic_launcher.png
|    |
|    ├──layout
|    |        activity_main.xml
|    |        fragment_main.xml
|    |
|    ├──menu
|    |        main.xml
|    |
|    ├──values
|    |        dimens.xml
|    |        strings.xml
|    |        styles.xml
|    |
|    ├──values-v11
|    |        styles.xml
|    |
|    ├──values-v14
|    |        styles.xml
|    |
|    └──values-w820dp
|            dimens.xml
|
```

```
└──src
    └──jp
        └──ac
            └──keio
                └──haruyamalabsns
                    │   MainActivity.java
                    │
                    ├──common
                    │       Constants.java
                    │       DbManager.java
                    │       FileUtility.java
                    │       Utility.java
                    │
                    ├──photo
                    │       CameraFragment.java
                    │       DataManager.java
                    │       TransferFileInfo.java
                    │
                    └──ref
                            MyNoticeService.java
                            UpdateReceiver.java
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="jp.ac.keio.haruyamalabsns"
    android:versionCode="1"
    android:versionName="1.0">

    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="21" />

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
    <uses-feature android:name="android.hardware.camera.flash" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:debuggable="true">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="landscape"
            android:launchMode="singleInstance">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="android.intent.category.BROWSABLE" />
            </intent-filter>
        </activity>

        <receiver
            android:name="org.altbeacon.beacon.startup.StartupBroadcastReceiver">
```

```
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <action android:name="android.intent.action.ACTION_POWER_CONNECTED" />
                <action
                    android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
            </intent-filter>
        </receiver>

        <service
            android:name="org.altbeacon.beacon.service.BeaconService"
            android:enabled="true"
            android:exported="false"
            android:isolatedProcess="false"
            android:label="beacon" />
        <service
            android:name="org.altbeacon.beacon.BeaconIntentProcessor"
            android:enabled="true"
            android:exported="false" />
        <service android:name="jp.ac.keio.haruyamalabsns.ref.MyNoticeService" />
    </application>

</manifest>
```

proguard-project.txt

```
# To enable ProGuard in your project, edit project.properties
# to define the proguard.config property as described in that file.
#
# Add project specific ProGuard rules here.
# By default, the flags in this file are appended to flags specified
# in ${sdk.dir}/tools/proguard/proguard-android.txt
# You can edit the include path and order by changing the ProGuard
# include property in project.properties.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# Add any project specific keep options here:

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}
```

project.properties

```
# This file is automatically generated by Android Tools.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file must be checked in Version Control Systems.
#
# To customize properties used by the Ant build system edit
# "ant.properties", and override values to adapt the script to your
# project structure.
#
# To enable ProGuard to shrink and obfuscate your code, uncomment this (available properties: sdk.dir, user.home):
#proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt

# Project target.
target=android-21
```

BuildConfig.java

```
/** Automatically generated file. DO NOT MODIFY */
package jp.ac.keio.haruyamalabsns;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package jp.ac.keio.haruyamalabsns;

public final class R {
    public static final class attr {
    }
    public static final class dimen {
        /**  Default screen margins, per the Android Design guidelines.

          Example customization of dimensions originally defined in res/values/dimens.xml
          (such as screen margins) for screens with more than 820dp of available width. This
          would include 7" and 10" devices in landscape (~960dp and ~1280dp respectively).

          */
        public static final int activity_horizontal_margin=0x7f040000;
        public static final int activity_vertical_margin=0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int action_settings=0x7f080009;
        public static final int button1=0x7f080003;
        public static final int button2=0x7f080002;
        public static final int button3=0x7f080005;
        public static final int button4=0x7f080007;
        public static final int button5=0x7f080006;
        public static final int container=0x7f080000;
        public static final int surface_view=0x7f080001;
        public static final int textView1=0x7f080004;
        public static final int textView3=0x7f080008;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
        public static final int fragment_main=0x7f030001;
    }
    public static final class menu {
        public static final int main=0x7f070000;
    }
    public static final class string {
        public static final int action_settings=0x7f050002;
        public static final int app_name=0x7f050000;
        public static final int hello_world=0x7f050001;
    }
    public static final class style {
        /**
        Base application theme, dependent on API level. This theme is replaced
        by AppBaseTheme from res/values-vXX/styles.xml on newer devices.


            Theme customizations available in newer API levels can go in
            res/values-vXX/styles.xml, while customizations related to
            backward-compatibility can go here.


        Base application theme for API 11+. This theme completely replaces
        AppBaseTheme from res/values/styles.xml on API 11+ devices.

 API 11 theme customizations can go here.

        Base application theme for API 14+. This theme completely replaces
```

```
        AppBaseTheme from BOTH res/values/styles.xml and
        res/values-v11/styles.xml on API 14+ devices.

  API 14 theme customizations can go here.
         */
        public static final int AppBaseTheme=0x7f060000;
        /**    Application theme.
  All customizations that are NOT specific to a particular API-level can go here.
         */
        public static final int AppTheme=0x7f060001;
    }
}
```

activity_main.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:context="jp.ac.keio.haruyamalabsns.MainActivity"
    tools:ignore="MergeRootFrame" >



</FrameLayout>
```

fragment_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="jp.ac.keio.haruyamalabsns.MainActivity$PlaceholderFragment" >

    <SurfaceView
        android:id="@+id/surface_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/button1"
        android:layout_alignBottom="@+id/button1"
        android:layout_alignParentRight="true"
        android:enabled="true"
        android:text="Off" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="22dp"
        android:layout_toLeftOf="@+id/button2"
        android:enabled="false"
        android:text="On" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button2"
        android:layout_alignParentRight="true"
        android:text="Location Information"
```

```
        android:textStyle="bold" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/button1"
        android:layout_alignBottom="@+id/button1"
        android:layout_alignParentLeft="true"
        android:enabled="false"
        android:text="Camera" />

    <Button
        android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignRight="@+id/button2"
        android:enabled="true"
        android:text="100m×100m" />

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/button5"
        android:layout_alignBottom="@+id/button5"
        android:layout_toLeftOf="@+id/button5"
        android:enabled="false"
        android:text="20m×20m" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignLeft="@+id/button3"
        android:layout_below="@+id/button3"
        android:layout_marginTop="28dp" />

</RelativeLayout>
```

main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="jp.ac.keio.haruyamalabsns.MainActivity" >

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/action_settings"/>

</menu>
```

dimens.xml

```
<resources>

    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>

</resources>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">HaruyamaLabSNS</string>
    <string name="action_settings">Settings</string>

</resources>
```

values¥styles.xml

```
<resources>

    <!--
        Base application theme, dependent on API level. This theme is replaced
        by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
    -->
    <style name="AppBaseTheme" parent="android:Theme.Light">
        <!--
            Theme customizations available in newer API levels can go in
            res/values-vXX/styles.xml, while customizations related to
            backward-compatibility can go here.
        -->
    </style>

    <!-- Application theme. -->
    <style name="AppTheme" parent="AppBaseTheme">
        <!-- All customizations that are NOT specific to a particular API-level can go here. -->
    </style>

</resources>
```

values-v11¥styles.xml

```
<resources>

    <!--
        Base application theme for API 11+. This theme completely replaces
        AppBaseTheme from res/values/styles.xml on API 11+ devices.
    -->
    <style name="AppBaseTheme" parent="android:Theme.Holo.Light">
        <!-- API 11 theme customizations can go here. -->
    </style>

</resources>
```

values-v14¥styles.xml

```
<resources>

    <!--
        Base application theme for API 14+. This theme completely replaces
        AppBaseTheme from BOTH res/values/styles.xml and
        res/values-v11/styles.xml on API 14+ devices.
    -->
    <style name="AppBaseTheme" parent="android:Theme.Holo.Light.DarkActionBar">
        <!-- API 14 theme customizations can go here. -->
    </style>

</resources>
```

dimens.xml

```
<resources>

    <!--
        Example customization of dimensions originally defined in res/values/dimens.xml
        (such as screen margins) for screens with more than 820dp of available width. This
        would include 7" and 10" devices in landscape (~960dp and ~1280dp respectively).
    -->
    <dimen name="activity_horizontal_margin">64dp</dimen>

</resources>
```

MainActivity.java(協生館検証前)

```
package jp.ac.keio.haruyamalabsns;

import java.util.HashMap;
import java.util.Properties;

import jp.ac.keio.haruyamalabsns.common.Utility;
import jp.ac.keio.haruyamalabsns.photo.CameraFragment;
import jp.ac.keio.haruyamalabsns.ref.MyNoticeService;
import jp.ac.keio.haruyamalabsns.ref.UpdateReceiver;

import org.altbeacon.beacon.Beacon;

import android.app.Fragment;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.PixelFormat;
import android.location.Location;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SurfaceView;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends FragmentActivity {

    private static final String TAG = "MainActivity";

    private Properties properties;

    private TextView textView;

    private boolean onLocationInfo;

    private boolean onUseCamera;

    private boolean isGPS100mArea;

    private String locationInformation;

    private UpdateReceiver upReceiver;

    private CameraFragment cameraFragment;

    private static PlaceholderFragment placeholderFragment;

    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        Log.d(TAG, "onCreate");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if (savedInstanceState == null) {
            placeholderFragment = new PlaceholderFragment();
            getFragmentManager().beginTransaction()
                    .add(R.id.container, placeholderFragment).commit();
        }

        init();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onRestart() {
        super.onRestart();
        setContentView(R.layout.activity_main);
        placeholderFragment = new PlaceholderFragment();
        getFragmentManager().beginTransaction().add(R.id.container, placeholderFragment)
                .commit();
        init();
    }

    @Override
    public void onPause() {
        super.onPause();
    }

    @Override
    public void onStop() {
        super.onStop();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        unregisterReceiver(upReceiver);
    }

    private Handler updateHandler = new Handler() {

        @Override
        public void handleMessage(Message msg) {
            Bundle bundle = msg.getData();
            String message = bundle.getString("message");
            Log.d(TAG, "Handler:" + message);

            textView = (TextView) findViewById(R.id.textView3);

            StringBuilder builder = new StringBuilder();
            if (!onLocationInfo) {
                return;
            }

            Location locationInfo = MyNoticeService.getLocationInfo();
```

```
            HashMap<String, Beacon> beacomMap = MyNoticeService.getBeacomMap();

            String hashtag = Utility.getHashtag(locationInfo, beacomMap, isGPS100mArea,
                    properties);
            locationInformation = hashtag;
            if (hashtag != null && !onUseCamera) {
                // Camera ボタン
                final Button buttonCamera = (Button) findViewById(R.id.button3);
                buttonCamera.setEnabled(true);
                onUseCamera = true;
            }

            if (hashtag == null) {
                hashtag = "";
            }

            builder.append("[hashtag]");
            builder.append("¥n");
            builder.append(hashtag);
            builder.append("¥n¥n");
            builder.append("[GPS Area]");
            builder.append("¥n");
            builder.append(Utility.getGPSAreaInfo(locationInfo, isGPS100mArea));
            builder.append("¥n¥n");
            builder.append("[Beacon]");
            builder.append("¥n");
            builder.append(Utility.getNearestBeaconsInfo(beacomMap));
            textView.setText(builder.toString());
        }
    };

    /**
     * A placeholder fragment containing a simple view.
     */
    public class PlaceholderFragment extends Fragment {

        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_main, container, false);

            // View 内の View 取得
            SurfaceView surfaceView = (SurfaceView) rootView
                    .findViewById(R.id.surface_view);
            surfaceView.setZOrderOnTop(true);
            surfaceView.getHolder().setFormat(PixelFormat.TRANSLUCENT);

            return rootView;
        }

        @Override
        public void onStart() {
            super.onStart();

            // On ボタン
            final Button buttonOn = (Button) findViewById(R.id.button1);
            // Off ボタン
            final Button buttonOff = (Button) findViewById(R.id.button2);
            // Camera ボタン
            final Button buttonCamera = (Button) findViewById(R.id.button3);
            // 10m2 ボタン
            final Button button10m2 = (Button) findViewById(R.id.button4);
            // 100m2 ボタン
            final Button button100m2 = (Button) findViewById(R.id.button5);

            buttonOn.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                startService(new Intent(MainActivity.this, MyNoticeService.class));
                buttonOn.setEnabled(false);
                buttonOff.setEnabled(true);
                buttonCamera.setEnabled(false);
                onLocationInfo = true;
                onUseCamera = false;
            }
        });

        buttonOff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                stopService(new Intent(MainActivity.this, MyNoticeService.class));
                buttonOn.setEnabled(true);
                buttonOff.setEnabled(false);
                buttonCamera.setEnabled(true);
                onLocationInfo = false;
                onUseCamera = true;
                locationInformation = null;
                if (textView != null) {
                    textView.setText("");
                }
            }
        });

        buttonCamera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Context context = MainActivity.this;
                cameraFragment = new CameraFragment(context, locationInformation);
                getSupportFragmentManager().beginTransaction()
                        .add(R.id.container, cameraFragment).commit();
            }
        });

        button10m2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                button10m2.setEnabled(false);
                button100m2.setEnabled(true);
                isGPS100mArea = false;
                MyNoticeService.setGPS100mAreaOnMainActivity(isGPS100mArea);
            }
        });

        button100m2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                button10m2.setEnabled(true);
                button100m2.setEnabled(false);
                isGPS100mArea = true;
                MyNoticeService.setGPS100mAreaOnMainActivity(isGPS100mArea);
            }
        });
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
    }

    @Override
    public void onPause() {
        super.onPause();
    }

    @Override
    public void onStop() {
```

```
            super.onStop();
        }

        @Override
        public void onDestroy() {
            super.onDestroy();
        }

    }

    public void init() {

        properties = Utility.loadBeaconProperties(getResources().getAssets());
        startService(new Intent(MainActivity.this, MyNoticeService.class));

        if (upReceiver != null) {
            unregisterReceiver(upReceiver);
        }
        upReceiver = new UpdateReceiver();
        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction("UPDATE_ACTION");
        registerReceiver(upReceiver, intentFilter);
        upReceiver.registerHandler(updateHandler);

        onLocationInfo = false;
        onUseCamera = true;
        isGPS100mArea = true;
        MyNoticeService.setGPS100mAreaOnMainActivity(isGPS100mArea);
    }

}
```

MainActivity.java(白川郷検証前)

```
package jp.ac.keio.haruyamalabsns;

import java.util.HashMap;
import java.util.Properties;

import jp.ac.keio.haruyamalabsns.common.Utility;
import jp.ac.keio.haruyamalabsns.photo.CameraFragment;
import jp.ac.keio.haruyamalabsns.ref.MyNoticeService;
import jp.ac.keio.haruyamalabsns.ref.UpdateReceiver;

import org.altbeacon.beacon.Beacon;

import android.app.Fragment;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.PixelFormat;
import android.location.Location;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SurfaceView;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends FragmentActivity {

    public static final String TAG = "MainActivity";
```

```
public static boolean onLocationInfo;

public static boolean onUseCamera;

public static String locationInformation;

public static TextView textView;

public static CameraFragment cameraFragment;

private Properties properties;

private boolean isGPS100mArea;

private UpdateReceiver upReceiver;

public static PlaceholderFragment placeholderFragment;

public MainActivity() {

}

@Override
public void onCreate(Bundle savedInstanceState) {
	Log.d(TAG, "onCreate");
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_main);
	if (savedInstanceState == null) {
		placeholderFragment = new PlaceholderFragment();
		getFragmentManager().beginTransaction()
				.add(R.id.container, placeholderFragment).commit();
	}

	init();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
	// Inflate the menu; this adds items to the action bar if it is present.
	getMenuInflater().inflate(R.menu.main, menu);
	return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
	int id = item.getItemId();
	if (id == R.id.action_settings) {
		return true;
	}
	return super.onOptionsItemSelected(item);
}

@Override
public void onRestart() {
	super.onRestart();
	setContentView(R.layout.activity_main);
	placeholderFragment = new PlaceholderFragment();
	getFragmentManager().beginTransaction().add(R.id.container, placeholderFragment)
			.commit();
	init();
}

@Override
public void onPause() {
	super.onPause();
}

@Override
public void onStop() {
	super.onStop();
```

```
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        unregisterReceiver(upReceiver);
    }

    private Handler updateHandler = new Handler() {

        @Override
        public void handleMessage(Message msg) {
            Bundle bundle = msg.getData();
            String message = bundle.getString("message");
            Log.d(TAG, "Handler:" + message);

            textView = (TextView) findViewById(R.id.textView3);

            StringBuilder builder = new StringBuilder();
            if (!onLocationInfo) {
                return;
            }

            Location locationInfo = MyNoticeService.getLocationInfo();
            HashMap<String, Beacon> beacomMap = MyNoticeService.getBeacomMap();

            String hashtag = Utility.getHashtag(locationInfo, beacomMap, isGPS100mArea,
                    properties);
            locationInformation = hashtag;
            if (hashtag != null && !onUseCamera) {
                // Camera ボタン
                final Button buttonCamera = (Button) findViewById(R.id.button3);
                buttonCamera.setEnabled(true);
                onUseCamera = true;
            }

            if (hashtag == null) {
                hashtag = "";
            }

            builder.append("[hashtag]");
            builder.append("¥n");
            builder.append(hashtag);
            builder.append("¥n¥n");
            builder.append("[GPS Area]");
            builder.append("¥n");
            builder.append(Utility.getGPSAreaInfo(locationInfo, isGPS100mArea));
            builder.append("¥n¥n");
            builder.append("[Beacon]");
            builder.append("¥n");
            builder.append(Utility.getNearestBeaconsInfo(beacomMap));
            textView.setText(builder.toString());
        }
    };

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment {

        public PlaceholderFragment() {
        }

        View rootView;

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
            rootView = inflater.inflate(R.layout.fragment_main, container, false);
```

```
        // View 内の View 取得
        SurfaceView surfaceView = (SurfaceView) rootView
                .findViewById(R.id.surface_view);
        surfaceView.setZOrderOnTop(true);
        surfaceView.getHolder().setFormat(PixelFormat.TRANSLUCENT);

        return rootView;
    }

    @Override
    public void onStart() {
        super.onStart();

        // On ボタン
        final Button buttonOn = (Button) rootView.findViewById(R.id.button1);
        // Off ボタン
        final Button buttonOff = (Button) rootView.findViewById(R.id.button2);
        // Camera ボタン
        final Button buttonCamera = (Button) rootView.findViewById(R.id.button3);

        buttonOn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Context context = getActivity();
                context.startService(new Intent(getActivity(), MyNoticeService.class));
                buttonOn.setEnabled(false);
                buttonOff.setEnabled(true);
                buttonCamera.setEnabled(false);
                onLocationInfo = true;
                onUseCamera = false;
            }
        });

        buttonOff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Context context = getActivity();
                context.stopService(new Intent(getActivity(), MyNoticeService.class));
                buttonOn.setEnabled(true);
                buttonOff.setEnabled(false);
                buttonCamera.setEnabled(true);
                onLocationInfo = false;
                onUseCamera = true;
                locationInformation = null;
                if (textView != null) {
                    textView.setText("");
                }
            }
        });

        buttonCamera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Context context = getActivity();
                cameraFragment = new CameraFragment(context, locationInformation);
                ((FragmentActivity) context).getSupportFragmentManager()
                        .beginTransaction().add(R.id.container, cameraFragment)
                        .commit();
            }
        });
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
    }

    @Override
    public void onPause() {
        super.onPause();
```

```
        }

        @Override
        public void onStop() {
            super.onStop();
        }

        @Override
        public void onDestroy() {
            super.onDestroy();
        }

    }

    public void init() {

        properties = Utility.loadBeaconProperties(getResources().getAssets());
        startService(new Intent(MainActivity.this, MyNoticeService.class));

        if (upReceiver != null) {
            unregisterReceiver(upReceiver);
        }
        upReceiver = new UpdateReceiver();
        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction("UPDATE_ACTION");
        registerReceiver(upReceiver, intentFilter);
        upReceiver.registerHandler(updateHandler);

        onLocationInfo = true;
        onUseCamera = false;
        isGPS100mArea = false;
        MyNoticeService.setGPS100mAreaOnMainActivity(isGPS100mArea);
    }

}
```

CameraFragment.java

```
package jp.ac.keio.haruyamalabsns.photo;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;

import jp.ac.keio.haruyamalabsns.R;
import jp.ac.keio.haruyamalabsns.common.Constants;
import jp.ac.keio.haruyamalabsns.common.FileUtility;
import jp.ac.keio.haruyamalabsns.common.Utility;

import org.apache.commons.net.ftp.FTPClient;

import android.app.AlertDialog;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Context;
import android.content.DialogInterface;
import android.graphics.PixelFormat;
import android.hardware.Camera;
import android.hardware.Camera.Size;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.provider.MediaStore.Images;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
```

```
import android.view.View;
import android.view.View.OnTouchListener;
import android.view.ViewGroup;
import android.view.ViewTreeObserver.OnGlobalLayoutListener;
import android.widget.EditText;

public class CameraFragment extends Fragment {
    private final static String TAG = "CameraFragment";

    private Camera camera_; // カメラインスタンス
    View rootView_; // ルート View
    SurfaceView surfaceView_; // プレビュー用 SurfaceView

    private boolean inPregress_ = false;

    private boolean isPassive = true;

    private Context context = null;

    private String locationInformation;

    private TransferFileInfo transferFileInfo = new TransferFileInfo();

    // Fragment コンストラクタ
    public CameraFragment(Context context, String locationInformation) {
        this.context = context;
        this.locationInformation = locationInformation;
    }

    // Surface リスナー
    private SurfaceHolder.Callback surfaceListener_ = new SurfaceHolder.Callback() {
        // Surface 作成
        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            // カメラインスタンスを取得
            camera_ = Camera.open();
            try {
                camera_.setPreviewDisplay(holder);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        // Surface 破棄時
        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            // カメラインスタンス開放
            if (camera_ != null) {
                camera_.release();
                camera_ = null;
            }
            holder.setFormat(PixelFormat.TRANSLUCENT);
        }

        // Surface 変更時
        // プレビューのパラメーターを設定し，プレビューを開始する
        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
            Log.d(TAG, "surfaceChanged width:" + width + " height:" + height);

            Camera.Parameters parameters = camera_.getParameters();

            // デバッグ用表示
            Size size = parameters.getPictureSize();
            Log.d(TAG, "getPictureSize width:" + size.width + " size.height:"
                    + size.height);
            size = parameters.getPreviewSize();
            Log.d(TAG, "getPreviewSize width:" + size.width + " size.height:"
                    + size.height);
```

```
            // 縦画面の場合回転させる
            if (rootView_.getWidth() < rootView_.getHeight()) {
                // 縦画面
                camera_.setDisplayOrientation(90);
            } else {
                // 横画面
                camera_.setDisplayOrientation(0);
            }

            List<Size> previewSizes = camera_.getParameters().getSupportedPreviewSizes();
            Size previewSize = previewSizes.get(0);
            parameters.setPreviewSize(previewSize.width, previewSize.height);
            // parameters.setPreviewSize(width, height);

            Log.d(TAG, "SupportedPreview.width:" + previewSize.width
                    + "SupportedPreview.height:" + previewSize.height);

            // パラメーターセット
            camera_.setParameters(parameters);
            // プレビュー開始
            camera_.startPreview();
        }
    };

    // シャッターが押されたときに呼ばれるコールバック
    private Camera.ShutterCallback shutterListener_ = new Camera.ShutterCallback() {
        public void onShutter() {
            final EditText editView = new EditText(context);
            AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(context);
            alertDialogBuilder.setIcon(android.R.drawable.ic_dialog_info);
            alertDialogBuilder.setTitle("Post to SNS");
            alertDialogBuilder.setView(editView);
            alertDialogBuilder.setPositiveButton("OK",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                            // コメントを取得
                            String comment = editView.getText().toString();

                            if ("".equals(comment.trim())) {
                                comment = null;
                            }

                            // 写真の転送
                            transferPicFile(transferFileInfo.getFilePicPath(),
                                    transferFileInfo.getJpgFileName());

                            // コメントテキストの転送
                            transferTextFile(transferFileInfo.getTxtFileName(), comment);
                        }
                    });
            alertDialogBuilder.setNegativeButton("Cancel",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {

                        }
                    });
            alertDialogBuilder.show();
        }
    };

    // JPEG イメージ生成後に呼ばれるコールバック
    private Camera.PictureCallback pictureListener_ = new Camera.PictureCallback() {
        // データ生成完了
        public void onPictureTaken(byte[] data, Camera camera) {

            String state = Environment.getExternalStorageState();
            if (!Environment.MEDIA_MOUNTED.equals(state)) {
                Log.d(TAG, "保存できません");
```

```
                return;
            }

            File rootDir = Environment
                    .getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
            if (!rootDir.exists()) {
                rootDir.mkdir();
            }

            String nowTime = Utility.getTime();
            String jpgFileName = String.format(Constants.JPG_FILE_NAME_FORMAT, nowTime);
            String txtFileName = String.format(Constants.TXT_FILE_NAME_FORMAT, nowTime);
            String filePath = String.format(Constants.FILE_PATH_FORMAT,
                    rootDir.getAbsolutePath(), jpgFileName);

            if (data != null) {

                saveImage(filePath, data);
                transferFileInfo = new TransferFileInfo();
                transferFileInfo.setFilePicPath(filePath);
                transferFileInfo.setJpgFileName(jpgFileName);
                transferFileInfo.setTxtFileName(txtFileName);

                // プレビューを再開する
                camera.startPreview();
                inPregress_ = false; // 処理中フラグをクリア
            }
        }
    };

    // 画面タッチ時のコールバック
    OnTouchListener ontouchListener_ = new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                if (camera_ != null && !inPregress_) {
                    // 撮影実行(AF 開始)
                    camera_.autoFocus(autoFocusListener_);
                }
            }
            return false;
        }
    };

    // AF 完了時のコールバック
    private Camera.AutoFocusCallback autoFocusListener_ = new Camera.AutoFocusCallback() {
        @Override
        public void onAutoFocus(boolean success, Camera camera) {
            inPregress_ = true; // 処理中フラグ
            camera.autoFocus(null);
            camera_.takePicture(shutterListener_, null, pictureListener_);
        }
    };

    Camera.ErrorCallback errorListener_ = new Camera.ErrorCallback() {
        @Override
        public void onError(int error, Camera camera) {
            camera_.stopPreview();
            camera_.release();
        }
    };

    // View 作成
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        // View 作成
        rootView_ = inflater.inflate(R.layout.fragment_main, container, false);
```

```
        // View 内の View 取得
        surfaceView_ = (SurfaceView) rootView_.findViewById(R.id.surface_view);

        // SurfaceHolder 設定
        SurfaceHolder holder = surfaceView_.getHolder();
        holder.addCallback(surfaceListener_);
        holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

        // タッチリスナー設定
        rootView_.setOnTouchListener(ontouchListener_);

        // 画面縦横比設定のため ViewTreeObserver にリスナー設定
        rootView_.getViewTreeObserver().addOnGlobalLayoutListener(
                new OnGlobalLayoutListener() {
                    // レイアウト完了時
                    @Override
                    public void onGlobalLayout() {
                    }
                });

        return rootView_;
    }

    @Override
    public void onDestroyView() {
        Log.d(TAG, "onDestroyView");
        super.onDestroyView();
    }

    @Override
    public void onPause() {
        Log.d(TAG, "onPause");
        super.onPause();
        if (camera_ != null) {
            camera_.stopPreview();
            camera_.release();
            camera_ = null;
        }
    }

    @Override
    public void onStop() {
        Log.d(TAG, "onStop");
        super.onStop();
    }

    @Override
    public void onDestroy() {
        Log.d(TAG, "onDestroy");
        super.onDestroy();
    }

    private void saveImage(String filePath, byte[] data) {
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(new File(filePath));
            fos.write(data);

            // アンドロイドのデータベースへの登録
            ContentValues values = new ContentValues();
            ContentResolver contentResolver = getActivity().getContentResolver();
            values.put(Images.Media.MIME_TYPE, "image/jpeg");
            values.put("_data", filePath);
            contentResolver.insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, values);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (fos != null) {
                    fos.close();
```

```
                    fos = null;
                }
            } catch (IOException e) {
                // 何もしない
            }
        }
    }

    private void transferPicFile(String filePath, String jpgFileName) {

        DataManager dataManager = new DataManager(getActivity(), filePath);
        dataManager.execute(Constants.ftpServerIp, Constants.ftpPort, jpgFileName,
                Constants.ftpId, Constants.ftpPass, isPassive, locationInformation,
                FTPClient.BINARY_FILE_TYPE);
    }

    private void transferTextFile(String txtFileName, String comment) {

        final String FORMAT_TEXT_PATH = "%s/%s";

        if (comment == null) {
            // 何もしない
            return;
        }

        FileUtility fileUtility = new FileUtility(context);
        try {
            fileUtility.createCommentFile(txtFileName, comment);

            File file = fileUtility.getFilesDir();
            String filePath = String.format(FORMAT_TEXT_PATH, file.getAbsolutePath(),
                    txtFileName);
            DataManager dataManager = new DataManager(getActivity(), filePath);
            dataManager.execute(Constants.ftpServerIp, Constants.ftpPort, txtFileName,
                    Constants.ftpId, Constants.ftpPass, isPassive, locationInformation,
                    FTPClient.ASCII_FILE_TYPE);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

DataManager.java

```
package jp.ac.keio.haruyamalabsns.photo;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTPReply;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.ContextWrapper;
import android.content.DialogInterface;
import android.os.AsyncTask;
import android.widget.Toast;

public class DataManager extends AsyncTask implements DialogInterface.OnCancelListener {

    private Context myContext;
    private String filePath;
    private ProgressDialog myProgressDialog;
    private FTPClient myFTPClient;

    public DataManager(Context context, String filePath) {
```

```
        this.myContext = context;
        this.filePath = filePath;
    }

    // 非同期処理開始
    @Override
    protected void onPreExecute() {
        myProgressDialog = new ProgressDialog(myContext);
        myProgressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        myProgressDialog.setCancelable(true);
        myProgressDialog.setOnCancelListener(this);
        myProgressDialog.setTitle("データ送信");
        myProgressDialog.setMessage("アップロード中");
        myProgressDialog.show();
    }

    // 非同期処理
    @Override
    protected Object doInBackground(Object... params) {
        String remoteserver = (String) params[0]; // FTP サーバードレス
        int remoteport = (Integer) params[1]; // FTP サーバーポート
        String remotefile = (String) params[2]; // 転送ファイル名
        String userid = (String) params[3]; // ログインユーザ ID
        String passwd = (String) params[4]; // ログインパスワード
        boolean passive = (Boolean) params[5]; // パッシブモード使用
        String uuidOrLocal = (String) params[6]; // beacon の UUID か位置情報
        int fileType = (Integer) params[7]; // 転送ファイルのタイプ

        // ＦＴＰファイル送信
        FTP ftp = new FTP(myContext);
        String result = ftp.putData(remoteserver, remoteport, userid, passwd, passive,
                remotefile, uuidOrLocal, fileType);
        ftp = null;

        return result;
    }

    // 非同期処理終了
    @Override
    protected void onPostExecute(Object result) {
        myProgressDialog.dismiss();
        if (result == null) {
            Toast.makeText(myContext, "データ送信終了", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(myContext, "データ送信 エラー発生", Toast.LENGTH_SHORT).show();
        }
    }

    // キャンセル処理
    @Override
    public void onCancel(DialogInterface dialog) {
        this.cancel(true);
    }

    @Override
    protected void onCancelled() {
        try {
            myFTPClient.abort();
        } catch (Exception e) {
        }
        Toast.makeText(myContext, "データ送信 キャンセル", Toast.LENGTH_SHORT).show();
    }

    // インナークラス　ＦＴＰクライアント commons net 使用
    private class FTP extends ContextWrapper {
        public FTP(Context base) {
            super(base);
        }
```

```
private String putData(String remoteserver, int remoteport, String userid,
        String passwd, boolean passive, String remotefile, String uuidOrLocal,
        int fileType) {
    int reply = 0;
    boolean isLogin = false;
    myFTPClient = new FTPClient();

    try {
        myFTPClient.setConnectTimeout(15000);
        // 接続
        myFTPClient.connect(remoteserver, remoteport);
        reply = myFTPClient.getReplyCode();
        if (!FTPReply.isPositiveCompletion(reply)) {
            throw new Exception("Connect Status:" + String.valueOf(reply));
        }
        // ログイン
        if (!myFTPClient.login(userid, passwd)) {
            throw new Exception("Invalid user/password");
        }
        isLogin = true;
        // 転送モード
        if (passive) {
            myFTPClient.enterLocalPassiveMode(); // パッシブモード
        } else {
            myFTPClient.enterLocalActiveMode(); // アクティブモード
        }
        myFTPClient.setFileType(fileType);

        if (!myFTPClient.changeWorkingDirectory(uuidOrLocal)) {
            if (!myFTPClient.makeDirectory(uuidOrLocal)) {
                throw new Exception("makeDirectory:error");
            }
            if (!myFTPClient.changeWorkingDirectory(uuidOrLocal)) {
                throw new Exception("changeWorkingDirectory:error");
            }
        }
        // ファイル送信
        myFTPClient.setDataTimeout(15000);
        myFTPClient.setSoTimeout(15000);
        FileInputStream fileInputStream = new FileInputStream(filePath);
        myFTPClient.storeFile(remotefile, fileInputStream);
        reply = myFTPClient.getReplyCode();
        if (!FTPReply.isPositiveCompletion(reply)) {
            throw new Exception("Send Status:" + String.valueOf(reply));
        }
        fileInputStream.close();
        fileInputStream = null;
        // ログアウト
        myFTPClient.logout();
        isLogin = false;
        // 切断
        myFTPClient.disconnect();
    } catch (Exception e) {
        return e.getMessage();
    } finally {
        if (isLogin) {
            try {
                myFTPClient.logout();
            } catch (IOException e) {
            }
        }
        if (myFTPClient.isConnected()) {
            try {
                myFTPClient.disconnect();
            } catch (IOException e) {
            }
        }
        myFTPClient = null;
```

```
                    if (fileType == FTPClient.ASCII_FILE_TYPE) {
                        // テキストファイルは残さないようにする。
                        File txtFile = new File(filePath);
                        txtFile.delete();
                    }
                }
                return null;
            }
        }

}
```

TransferFileInfo.java

```
package jp.ac.keio.haruyamalabsns.photo;

public class TransferFileInfo {

    private String filePicPath;

    private String fileTextPath;

    private String jpgFileName;

    private String txtFileName;

    private String comment;

    public String getFilePicPath() {
        return filePicPath;
    }

    public void setFilePicPath(String filePicPath) {
        this.filePicPath = filePicPath;
    }

    public String getFileTextPath() {
        return fileTextPath;
    }

    public void setFileTextPath(String fileTextPath) {
        this.fileTextPath = fileTextPath;
    }

    public String getJpgFileName() {
        return jpgFileName;
    }

    public void setJpgFileName(String jpgFileName) {
        this.jpgFileName = jpgFileName;
    }

    public String getTxtFileName() {
        return txtFileName;
    }

    public void setTxtFileName(String txtFileName) {
        this.txtFileName = txtFileName;
    }

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }

}
```

MyNoticeService.java

```
package jp.ac.keio.haruyamalabsns.ref;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Properties;
import java.util.Timer;
import java.util.TimerTask;

import jp.ac.keio.haruyamalabsns.common.Constants;
import jp.ac.keio.haruyamalabsns.common.DbManager;
import jp.ac.keio.haruyamalabsns.common.Utility;

import org.altbeacon.beacon.Beacon;
import org.altbeacon.beacon.BeaconManager;
import org.altbeacon.beacon.BeaconParser;
import org.altbeacon.beacon.RangeNotifier;
import org.altbeacon.beacon.Region;
import org.altbeacon.beacon.startup.BootstrapNotifier;
import org.altbeacon.beacon.startup.RegionBootstrap;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.RemoteException;
import android.util.Log;

public class MyNoticeService extends Service implements BootstrapNotifier {

	private final static String TAG = "MyNoticeService";

	private static HashMap<String, Beacon> beacomMap = new HashMap<String, Beacon>();

	private static Location locationInfo;

	private Properties properties;

	private RegionBootstrap regionBootstrap;

	private LocationManager locationManager;

	private BeaconManager beaconManager;

	private Handler handler;

	private Timer timer = new Timer();

	private DbManager dbManager = DbManager.getInstance();

	private List<String> hashtags = new ArrayList<String>();

	private static boolean isGPS100mAreaOnMainActivity;

	private LocationListener[] mLocationListeners = new LocationListener[] {
			new LocationListener(LocationManager.GPS_PROVIDER),
			new LocationListener(LocationManager.NETWORK_PROVIDER),
			new LocationListener(LocationManager.PASSIVE_PROVIDER) };

	private class LocationListener implements android.location.LocationListener {

		public LocationListener(String provider) {
```

```
            Log.e(TAG, "LocationListener " + provider);
            locationInfo = new Location(provider);
        }

        @Override
        public void onLocationChanged(Location location) {
            Log.e(TAG, "onLocationChanged: " + location);
            setLocationInfo(location);
        }

        @Override
        public void onProviderDisabled(String provider) {
            Log.e(TAG, "onProviderDisabled: " + provider);
        }

        @Override
        public void onProviderEnabled(String provider) {
            Log.e(TAG, "onProviderEnabled: " + provider);
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            Log.e(TAG, "onStatusChanged: " + provider);
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        Log.i(TAG, "onBind" + ": " + intent);
        return null;
    }

    @Override
    public void onRebind(Intent intent) {
        Log.i(TAG, "onRebind" + ": " + intent);
    }

    @Override
    public boolean onUnbind(Intent intent) {
        Log.i(TAG, "onUnbind" + ": " + intent);
        return true;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d(TAG, "onCreate");

        properties = Utility.loadBeaconProperties(getResources().getAssets());

        // ロケーションマネージャのインスタンスを取得する
        locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

        // 位置情報の更新を受け取るように設定
        if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
                    mLocationListeners[0]);
        } else if (locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {
            locationManager.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER, 0,
                    0, mLocationListeners[1]);
        } else {
            locationManager.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER, 0,
                    0, mLocationListeners[2]);
        }

        // iBeacon のデータを受信できるように Parser を設定
        beaconManager = BeaconManager.getInstanceForApplication(this);
        beaconManager.getBeaconParsers().add(
                new BeaconParser().setBeaconLayout(Constants.IBEACON_FORMAT));
        // BG で iBeacon 領域を監視(モニタリング)するスキャン間隔を設定
```

```
        beaconManager.setBackgroundScanPeriod(3000);
        beaconManager.setForegroundScanPeriod(3000);
        beaconManager.setBackgroundBetweenScanPeriod(3000);
        beaconManager.setForegroundBetweenScanPeriod(3000);
        // UUID, major, minor の指定はしない
        Region region = new Region("ibeacon", null, null, null);
        regionBootstrap = new RegionBootstrap(this, region);

        beaconManager.addRangeNotifier(new RangeNotifier() {
            @Override
            public void didRangeBeaconsInRegion(Collection<Beacon> beacons, Region region) {
                // 検出したビーコンの情報を全部 Log に書き出す
                String newUUIDInfo = null;
                double distance = 0.0;
                for (Beacon beacon : beacons) {
                    newUUIDInfo = String.format(Constants.UUID_FORMAT, beacon.getId1()
                            .toString(), beacon.getId2().toString(), beacon.getId3()
                            .toString());
                    // 大文字で統一
                    newUUIDInfo = newUUIDInfo.toUpperCase();
                    distance = beacon.getDistance();
                    beacomMap.put(newUUIDInfo, beacon);
                    Log.d(TAG,
                            "UUID:" + beacon.getId1() + ", major:" + beacon.getId2()
                                    + ", minor:" + beacon.getId3() + ", Distance:"
                                    + distance + ",RSSI" + beacon.getRssi() + ", TxPower"
                                    + beacon.getTxPower());
                }
            }
        });

        beaconManager.setRangeNotifier(new RangeNotifier() {
            @Override
            public void didRangeBeaconsInRegion(Collection<Beacon> beacons, Region region) {

                // 検出したビーコンの情報を全部みる
                String newUUIDInfo = null;
                double distance = Double.MAX_VALUE;
                for (Beacon beacon : beacons) {
                    newUUIDInfo = String.format(Constants.UUID_FORMAT, beacon.getId1()
                            .toString(), beacon.getId2().toString(), beacon.getId3()
                            .toString());
                    // 大文字で統一
                    newUUIDInfo = newUUIDInfo.toUpperCase();
                    distance = beacon.getDistance();
                    beacomMap.put(newUUIDInfo, beacon);
                    Log.d(TAG,
                            "UUID:" + beacon.getId1() + ", major:" + beacon.getId2()
                                    + ", minor:" + beacon.getId3() + ", Distance:"
                                    + distance + ",RSSI" + beacon.getRssi() + ", TxPower"
                                    + beacon.getTxPower());
                }
            }
        });
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(TAG, "onStartCommand");

        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                StringBuilder message = new StringBuilder();
                // if (mLastLocation != null && (int)
                // (mLastLocation.getLatitude()) != 0) {
                // // 緯度
                // message.append("緯度=");
                // message.append(mLastLocation.getLatitude());
```

```
                    // message.append(",");
                    // // 経度
                    // message.append("経度=");
                    // message.append(mLastLocation.getLongitude());
                    // message.append("¥n");
                    // }
                    // // 一番距離が近いビーコン情報を取得
                    // message.append(getNearestBeaconsInfo());
                    // if ("".equals(message.toString())) {
                    // message.append("位置情報取得中…");
                    // }
                    // 空メッセージを送信
                    sendBroadCast(message.toString());

                    noticeURL();

                }
            }, 0, Constants.INTERVAL_PERIOD);

            return START_STICKY;
        }

        @Override
        public void onDestroy() {
            Log.d(TAG, "onDestroy");
            super.onDestroy();
            if (locationManager != null) {
                for (int i = 0; i < mLocationListeners.length; i++) {
                    try {
                        locationManager.removeUpdates(mLocationListeners[i]);
                    } catch (Exception ex) {
                        Log.i(TAG, "fail to remove location listener, ignore", ex);
                    }
                }
            }
            locationInfo.reset();
            beacomMap.clear();
            hashtags.clear();
            sendBroadCast("");
            if (timer != null) {
                timer.cancel();
            }
        }

        public void registerHandler(Handler UpdateHandler) {
            handler = UpdateHandler;
        }

        protected void sendBroadCast(String message) {

            Intent broadcastIntent = new Intent();
            broadcastIntent.putExtra("message", message);
            broadcastIntent.setAction("UPDATE_ACTION");
            getBaseContext().sendBroadcast(broadcastIntent);

        }

        @Override
        public void didDetermineStateForRegion(int i, Region region) {
            // 入退場状態が変更された
            Log.d(TAG, "Determine State: " + i);
        }

        @Override
        public void didEnterRegion(Region region) {
            // 領域に入場した
            Log.d(TAG, "Enter Region");
            try {
                // レンジング開始
```

```
            beaconManager.startRangingBeaconsInRegion(region);
        } catch (RemoteException e) {
            // 例外が発生した場合
            e.printStackTrace();
        }
    }

    @Override
    public void didExitRegion(Region region) {
        // 領域から退場した
        Log.d(TAG, "Exit Region");
        // レンジング停止
        try {
            beaconManager.stopRangingBeaconsInRegion(region);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

    public static HashMap<String, Beacon> getBeacomMap() {
        return beacomMap;
    }

    public static Location getLocationInfo() {
        return locationInfo;
    }

    public static void setLocationInfo(Location locationInfo) {
        MyNoticeService.locationInfo = locationInfo;
    }

    public static boolean isGPS100mAreaOnMainActivity() {
        return isGPS100mAreaOnMainActivity;
    }

    public static void setGPS100mAreaOnMainActivity(boolean isGPS100mAreaOnMainActivity) {
        MyNoticeService.isGPS100mAreaOnMainActivity = isGPS100mAreaOnMainActivity;
    }

    private void noticeURL() {
        String hashtag = Utility.getHashtag(locationInfo, beacomMap,
                isGPS100mAreaOnMainActivity, properties);
        if (hashtag == null) {
            return;
        }

        if (hashtags.contains(hashtag)) {
            return;
        }

        hashtags.add(hashtag);

        try {
            dbManager.openConnection();
            if (dbManager.isExistsHashtag(hashtag)) {
                String contentText = String.format(Constants.URL_FORMAT_WITH_HASHTAG,
                        hashtag);
                Uri uri = Uri.parse(contentText);
                Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            }
        } catch (Exception e) {
            // DB の例外は無視
            Log.d(TAG, "noticeURL:" + e.getMessage());
        } finally {
            try {
                dbManager.closeConnection();
            } catch (SQLException e) {
                // DB の例外は無視
```

```
                    Log.d(TAG, "noticeURL:" + e.getMessage());
                }
            }
        }
    }
```

UpdateReceiver.java

```
package jp.ac.keio.haruyamalabsns.ref;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;

public class UpdateReceiver extends BroadcastReceiver {

    public static Handler handler;

    @Override
    public void onReceive(Context context, Intent intent) {

        Bundle bundle = intent.getExtras();
        String message = bundle.getString("message");

        if (handler != null) {
            Message msg = new Message();

            Bundle data = new Bundle();
            data.putString("message", message);
            msg.setData(data);
            handler.sendMessage(msg);
        }
    }

    /**
     * メイン画面の表示を更新
     */
    public void registerHandler(Handler locationUpdateHandler) {
        handler = locationUpdateHandler;
    }
}
```

Constants.java

```
package jp.ac.keio.haruyamalabsns.common;

public class Constants {

    public static final String SERVER_HOST = "xx.xx.xx.xx";

    public static final String ftpServerIp = "xx.xx.xx.xx";

    public static final int ftpPort = 21;

    public static final String ftpId = "xxx";

    public static final String ftpPass = "xxx";

    public static final String URL_FORMAT = "http://" + SERVER_HOST
            + ":8080/Torafugu/TorafuguServlet";

    public static final String URL_FORMAT_WITH_HASHTAG = "http://" + SERVER_HOST
            + ":8080/Torafugu/TorafuguServlet?hashtag=%s";

    public static final String LOCATION_INFO_FORMAT = "Longitude(経度)=%s, Latitude(緯度)=%s";

    public static final String STATUS_FORMAT = "【LocationButton】: %s, 【GPS100m2Button】:%s";
```

```
    public static final String DB_URL = "jdbc:postgresql://" + SERVER_HOST + ":5432/HaruyamaLab";

    public static final String DB_USER = "HaruyamaLab";

    public static final String DB_PASSWORD = "HaruyamaLab";

    public static final String SELECT_COUNT_HASHTAG_FORMAT = "SELECT COUNT(*) AS cnthash FROM
HASHTAG WHERE hashtag=?";

    public static final String SELECT_HASHTAG_FROM_BEACON_FORMAT = "SELECT hashtag FROM
BEACON WHERE uuid=?";

    public static final int PREVIEW_WIDTH = 1350;

    public static final int PREVIEW_HEIGHT = 1080;

    // iBeacon のデータを認識するための Parser フォーマット
    public static final String IBEACON_FORMAT = "m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24";

    public static final String UUID_FORMAT = "%s_%s_%s";

    public static final String GPS_AREA_FORMAT = "%s_%s";

    public static final String HASHTAG_FORMAT = "KeioSDMHaruyamaLab_%s";

    public static final String JPG_FILE_NAME_FORMAT = "%s.jpg";

    public static final String TXT_FILE_NAME_FORMAT = "%s.txt";

    public static final String FILE_PATH_FORMAT = "%s/%s";

    public static final int INTERVAL_PERIOD = 10000;

}
```

DbManager.java

```
package jp.ac.keio.haruyamalabsns.common;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;

import android.util.Log;

public class DbManager {

    private final static String TAG = "DbManager";

    public static DbManager dbManager = new DbManager();

    private Connection conn = null;

    private DbManager() {

    }

    public static DbManager getInstance() {
        return dbManager;
    }

    public void openConnection() throws ClassNotFoundException, SQLException {
        if (conn != null) {
            return;
        }
        Class.forName("org.postgresql.Driver");
```

```
            Properties props = new Properties();
            props.setProperty("user", Constants.DB_USER);
            props.setProperty("password", Constants.DB_PASSWORD);
            props.setProperty("ssl", "false");
            conn = DriverManager.getConnection(Constants.DB_URL, props);
      }

      public String getHashTag(String uuid) throws SQLException {
            Log.d(TAG, "uuid=" + uuid);
            conn.setAutoCommit(false);
            PreparedStatement ps = conn
                        .prepareStatement(Constants.SELECT_HASHTAG_FROM_BEACON_FORMAT);
            ps.setString(1, uuid);
            ResultSet rs = ps.executeQuery();
            rs.next();
            String hashtag = rs.getString("hashtag");
            Log.d(TAG, "hashtag=" + hashtag);
            return hashtag;
      }

      public boolean isExistsHashtag(String hashtag) throws SQLException {
            Log.d(TAG, "isExistsHashtag:" + hashtag);
            boolean exists = false;

            conn.setAutoCommit(false);
            PreparedStatement ps = conn
                        .prepareStatement(Constants.SELECT_COUNT_HASHTAG_FORMAT);
            ps.setString(1, hashtag);
            ResultSet rs = ps.executeQuery();
            rs.next();
            int num = rs.getInt("cnthash");
            if (num > 0) {
                  exists = true;
            }
            return exists;
      }

      public void closeConnection() throws SQLException {
            if (conn != null) {
                  conn.close();
                  conn = null;
            }
      }
}
```

FileUtility.java

```
package jp.ac.keio.haruyamalabsns.common;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;

import android.content.Context;
import android.content.ContextWrapper;

public class FileUtility extends ContextWrapper {

      public FileUtility(Context base) {
            super(base);
      }

      public void createCommentFile(String fileName, String comment) throws IOException {
            OutputStream out = null;
            PrintWriter writer = null;
            try {
                  out = openFileOutput(fileName, MODE_PRIVATE);
                  writer = new PrintWriter(new OutputStreamWriter(out, "UTF-8"));
```

```
                // writer = new PrintWriter(new OutputStreamWriter(out));
                // 追記する
                writer.append(comment);
            } finally {
                if (writer != null) {
                    writer.close();
                }
            }
        }

        public void deleteCommentFile(String commentFile) {
            deleteFile(commentFile);
        }

        public boolean exists(String fileName) {
            File file = getFileStreamPath(fileName);
            return file.exists();
        }
}
```

Utility.java

```
package jp.ac.keio.haruyamalabsns.common;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Properties;

import org.altbeacon.beacon.Beacon;

import android.content.res.AssetManager;
import android.location.Location;
import android.util.Log;

public class Utility {

    private final static String TAG = "DbManager";

    public static Properties properties;

    public static boolean existsLoaction(Location locationInfo,
            HashMap<String, Beacon> beacomMap) {
        boolean existsLoaction = false;

        if (!((int) locationInfo.getLongitude() == 0 && (int) locationInfo.getLatitude() == 0)) {
            existsLoaction = true;
        }

        if (!beacomMap.isEmpty()) {
            existsLoaction = true;
        }

        return existsLoaction;

    }

    public static String getAllBeaconsInfo(HashMap<String, Beacon> beacomMap) {
        StringBuilder builder = new StringBuilder();
        for (String key : beacomMap.keySet()) {
            Beacon beacon = beacomMap.get(key);
            int rssi = beacon.getRssi();
            int txPower = beacon.getTxPower();
            builder.append("¥n");
            builder.append(" UUID_Major_Minor=" + key);
            builder.append("¥n");
```

```
            builder.append(" Distance=" + Double.toString(beacon.getDistance()));
            builder.append("¥n");
            builder.append(" Rssi=" + Integer.toString(rssi));
            builder.append("¥n");
            builder.append(" BeaconTypeCode="
                    + Integer.toString(beacon.getBeaconTypeCode()));
            builder.append("¥n");
            builder.append(" BluetoothAddress=" + beacon.getBluetoothAddress());
            builder.append("¥n");
            builder.append(" BluetoothName=" + beacon.getBluetoothName());
            builder.append("¥n");
            builder.append(" Manufacturer=" + Integer.toString(beacon.getManufacturer()));
            builder.append("¥n");
            builder.append(" ParserIdentifier=" + beacon.getParserIdentifier());
            builder.append("¥n");
            builder.append(" TxPower=" + txPower);
            builder.append("¥n");
            builder.append(" Proximity=" + Integer.toString(10 ^ ((txPower - rssi) / 20)));
            builder.append("¥n");
        }
        return builder.toString();
    }

    public static String getGPSAreaInfo(Location locationInfo, boolean isGPS100mArea) {

        // 緯度
        double longitude = roundingOffbyGPSArea(locationInfo.getLongitude(),
                isGPS100mArea);

        // 経度
        double latitude = roundingOffbyGPSArea(locationInfo.getLatitude(), isGPS100mArea);

        String locaionInfo = String.format(Constants.LOCATION_INFO_FORMAT, longitude,
                latitude);
        return locaionInfo;
    }

    public static String getHashtag(Location locationInfo,
            HashMap<String, Beacon> beacomMap, boolean isGPS100mArea,
            Properties properties) {

        String hashtag = null;
        String value = getLongitudeAndLatitude(locationInfo, isGPS100mArea);
        if (value != null) {
            hashtag = String.format(Constants.HASHTAG_FORMAT, value);
            return hashtag;
        }

        String uuid = getNearestUUID(beacomMap);
        if ("UNKNOWN".equals(uuid)) {
            return hashtag;
        }

        try {
            String converedHashtag = properties.getProperty(uuid);
            if (converedHashtag != null) {
                hashtag = String.format(Constants.HASHTAG_FORMAT, converedHashtag);
            } else {
                hashtag = String.format(Constants.HASHTAG_FORMAT, uuid.replaceAll("-", "_"));
            }
        } catch (Exception e) {
            // DB の例外は無視
           Log.d(TAG, "getHashtag:" + e.getMessage());
        }

        return hashtag;
    }

    public static String getLoactionInfo(Location locationInfo,
            HashMap<String, Beacon> beacomMap, boolean isGPS100mArea) {
```

```
        String locationInformation;
        if (!((int) locationInfo.getLongitude() == 0 && (int) locationInfo.getLatitude() == 0)) {
            locationInformation = getLongitudeAndLatitude(locationInfo, isGPS100mArea);
        } else {
            locationInformation = getNearestUUID(beacomMap);
        }

        return locationInformation;
    }

    public static String getLongitudeAndLatitude(Location locationInfo,
            boolean isGPS100mArea) {

        String longitudeAndLatitude = null;
        if (!((int) locationInfo.getLongitude() == 0 && (int) locationInfo.getLatitude() == 0)) {

            // 緯度
            double longitude = roundingOffbyGPSArea(locationInfo.getLongitude(),
                    isGPS100mArea);

            // 経度
            double latitude = roundingOffbyGPSArea(locationInfo.getLatitude(),
                    isGPS100mArea);

            String value = String.format(Constants.GPS_AREA_FORMAT,
                    String.valueOf(longitude), String.valueOf(latitude));
            // .はハッシュタグとして認識されないため別の文字列に置換
            longitudeAndLatitude = value.replaceAll("¥¥.", "point");
        }

        return longitudeAndLatitude;
    }

    public static String getNearestUUID(HashMap<String, Beacon> beacomMap) {
        String NearestUUID = "UNKNOWN";
        double distance = Double.MAX_VALUE;
        for (String key : beacomMap.keySet()) {
            Beacon beacon = beacomMap.get(key);
            double distance2 = beacon.getDistance();
            if (distance2 < distance) {
                NearestUUID = key;
                distance = distance2;
            }
        }
        return NearestUUID.toUpperCase();
    }

    public static String getNearestBeaconsInfo(HashMap<String, Beacon> beacomMap) {
        StringBuilder builder = new StringBuilder();
        String nearestKey = null;
        double distance = Double.MAX_VALUE;
        for (String key : beacomMap.keySet()) {
            Beacon beacon = beacomMap.get(key);
            double distance2 = beacon.getDistance();
            if (distance2 < distance) {
                nearestKey = key;
                distance = distance2;
            }
        }

        if (nearestKey == null) {
            return builder.toString();
        }

        Beacon beacon = beacomMap.get(nearestKey);
        int rssi = beacon.getRssi();
        int txPower = beacon.getTxPower();
        builder.append(" UUID_Major_Minor=" + nearestKey);
        builder.append("¥n");
        builder.append(" Distance=" + Double.toString(beacon.getDistance()));
```

```
            builder.append("¥n");
            builder.append(" Rssi=" + Integer.toString(rssi));
            builder.append("¥n");
            builder.append(" BeaconTypeCode=" + Integer.toString(beacon.getBeaconTypeCode()));
            builder.append("¥n");
            builder.append(" BluetoothAddress=" + beacon.getBluetoothAddress());
            builder.append("¥n");
            builder.append(" BluetoothName=" + beacon.getBluetoothName());
            builder.append("¥n");
            builder.append(" Manufacturer=" + Integer.toString(beacon.getManufacturer()));
            builder.append("¥n");
            builder.append(" ParserIdentifier=" + beacon.getParserIdentifier());
            builder.append("¥n");
            builder.append(" TxPower=" + txPower);
            builder.append("¥n");
            builder.append(" Proximity=" + Integer.toString(10 ^ ((txPower - rssi) / 20)));
            builder.append("¥n");

            return builder.toString();
        }

        public static String getTime() {
            final String TIME_FORMAT = "%04d%02d%02d_%02d%02d%03d";
            String now = null;
            Date date = new Date();
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(date);

            now = String.format(TIME_FORMAT, calendar.get(Calendar.YEAR),
                    calendar.get(Calendar.MONTH) + 1, calendar.get(Calendar.DAY_OF_MONTH),
                    calendar.get(Calendar.HOUR_OF_DAY), calendar.get(Calendar.MINUTE),
                    calendar.get(Calendar.SECOND));
            return now;
        }

        public static Properties loadBeaconProperties(AssetManager as) {
            String filePath = "beacon.properties";
            try {
                if (properties != null) {
                    return properties;
                } else {
                    properties = new Properties();
                }
                properties.load(as.open(filePath));
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return properties;
        }

        public static double roundingOffbyGPSArea(double location, boolean isGPS100mArea) {

            BigDecimal bdLatitude = new BigDecimal(location);

            double latitude;
            if (!isGPS100mArea) {
                // 小数点第五位を四捨五入
                latitude = bdLatitude.setScale(4, BigDecimal.ROUND_HALF_UP).doubleValue();
            } else {
                // 小数点第四位を四捨五入
                latitude = bdLatitude.setScale(3, BigDecimal.ROUND_HALF_UP).doubleValue();
            }
            return latitude;
        }

}
```

11.1.2. サーバー

(1) Torafugu プロジェクト

プロジェクト構成例

```
root フォルダ
 |
 └──WEB-INF
      |   web.xml
      |
      ├──lib
      |       commons-codec-1.8.jar
      |       commons-lang3-3.1.jar
      |       commons-logging-1.2.jar
      |       gson-2.2.2.jar
      |       jInstagram-1.2.1.jar
      |       junit-4.11.jar
      |       mockito-all-1.8.4.jar
      |       slf4j-api-1.7.5.jar
      |
      └──src
          └──jp
              └──ac
                  └──keio
                      └──haruyamalabsns
                              Constants.java
                              TorafuguService.java
                              TorafuguServlet.java
```

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0"
    metadata-complete="true">

    <description>
        keio haruyamalab snsServlet.
    </description>
    <display-name>Torafugu</display-name>

    <servlet>
        <servlet-name>TorafuguServlet</servlet-name>
        <servlet-class>jp.ac.keio.haruyamalabsns.TorafuguServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>TorafuguService</servlet-name>
        <servlet-class>jp.ac.keio.haruyamalabsns.TorafuguService</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>TorafuguServlet</servlet-name>
        <url-pattern>/TorafuguServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>TorafuguService</servlet-name>
        <url-pattern>/TorafuguService</url-pattern>
    </servlet-mapping>

</web-app>
```

TorafuguService.java

```
package jp.ac.keio.haruyamalabsns;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;
import java.util.List;
import java.util.Map;
import java.util.TimeZone;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.jinstagram.Instagram;
import org.jinstagram.InstagramClient;
import org.jinstagram.auth.InstagramAuthService;
import org.jinstagram.auth.model.Token;
import org.jinstagram.auth.model.Verifier;
import org.jinstagram.auth.oauth.InstagramService;
import org.jinstagram.entity.comments.CommentData;
import org.jinstagram.entity.common.Caption;
import org.jinstagram.entity.common.Comments;
import org.jinstagram.entity.common.ImageData;
import org.jinstagram.entity.common.Images;
import org.jinstagram.entity.common.Likes;
import org.jinstagram.entity.common.Location;
import org.jinstagram.entity.users.feed.MediaFeed;
import org.jinstagram.entity.users.feed.MediaFeedData;
import org.jinstagram.exceptions.InstagramException;

@WebServlet("/TorafuguService")
public class TorafuguService extends HttpServlet {

	private static final long serialVersionUID = 1L;

	public static int count = 0;

	@Override
	public void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {

		HttpSession httpSession = null;
		String hashtag = null;
		try {
			response.setContentType("text/html; charset=UTF-8");

			httpSession = request.getSession();

			Object sessionObj = httpSession.getAttribute("hashtag");
			if (sessionObj != null) {
				hashtag = (String) sessionObj;
			}

			Map parameterMap = request.getParameterMap();
			String par[] = (String[]) parameterMap.get("code");

			String verifierCode = par[0];

			InstagramService service = new InstagramAuthService()
					.apiKey(Constants.ACCESS_ID).apiSecret(Constants.ACCESS_SECRET)
					.scope("public_content")
					// .scope("basic public_content follower_list relationships")
					.callback(Constants.SERVER_URL2).build();

			Verifier verifier = new Verifier(verifierCode);
			Token accessToken = service.getAccessToken(verifier);
```

```
            InstagramClient instagram = new Instagram(accessToken);
            MediaFeed mediaFeed;
            if (hashtag != null) {
                // ハッシュタグで検索
                mediaFeed = instagram.getRecentMediaFeedTags(hashtag);
            } else {
                /* 自分のアカウントから画像を取得 */
                mediaFeed = instagram.getUserRecentMedia();
            }

            doService(response, instagram, mediaFeed, hashtag);

        } catch (InstagramException e) {
            e.printStackTrace();
        } finally {
            if (httpSession != null) {
                // セッション終了
                httpSession.invalidate();
            }
        }
    }

    private void doService(HttpServletResponse response, InstagramClient instagram,
            MediaFeed mediaFeed, String hashtag) throws IOException {
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>HaruyamaLab SNS</title>");
        out.println("</head>");
        out.println("<body>");

        List<MediaFeedData> mediaFeeds = mediaFeed.getData();
        if (mediaFeeds.isEmpty()) {
            out.println("<p>hashtag=" + hashtag + "</p>");
            out.println("<p>投稿写真はありません。</p>");
        }

        int i = 0;
        for (MediaFeedData mediaData : mediaFeeds) {
            out.println("<div style=¥"float:left; margin-left:20px¥">");
            out.println("<h1>写真" + (i + 1) + "</h1>");

            Calendar createdTime = Calendar.getInstance();
            createdTime.setTimeZone(TimeZone.getTimeZone("Asia/Tokyo"));
            /* 作成日時をエポックミリ秒で扱う */
            String createdTimeStr = mediaData.getCreatedTime() + "000";
            long time = Long.parseLong(createdTimeStr);
            createdTime.setTimeInMillis(time);
            createdTime.add(Calendar.MONTH, 1);
            System.out.print(createdTime.get(Calendar.YEAR) + "/");
            System.out.print(createdTime.get(Calendar.MONTH) + "/");
            System.out.print(createdTime.get(Calendar.DAY_OF_MONTH) + " ");
            System.out.print(createdTime.get(Calendar.HOUR_OF_DAY) + ":");
            System.out.print(createdTime.get(Calendar.SECOND));
            out.print("<p>【投稿日】");
            out.print(createdTime.get(Calendar.YEAR) + "/");
            out.print(createdTime.get(Calendar.MONTH) + "/");
            out.print(createdTime.get(Calendar.DAY_OF_MONTH) + " ");
            out.print(createdTime.get(Calendar.HOUR_OF_DAY) + ":");
            out.print(createdTime.get(Calendar.SECOND));
            out.println("</p>");

            Images images = mediaData.getImages();
            ImageData imageData = images.getLowResolution();
            String imageUrl = imageData.getImageUrl();
            int jpgIndex = imageUrl.indexOf(".jpg?");
            if (jpgIndex != -1) {
```

```
                imageUrl = imageUrl.substring(0, jpgIndex + 4);
            }

            out.println("<img src=¥"" + imageUrl + "¥">");

            Caption caption = mediaData.getCaption();
            if (caption != null) {
                String captionText = caption.getText();
                if (captionText != null && !captionText.trim().equals("")) {
                    out.println("<p>" + captionText.replaceAll("¥n", "<br>") + "</p>");
                    System.out.println(captionText);
                }
            }

            System.out.println("### CreatedTime ###");
            System.out.println(mediaData.getCreatedTime());

            double latitude = 0.0;
            double longitude = 0.0;
            Location location = mediaData.getLocation();
            // if (location != null) {
            // String locationName = location.getName();
            // if (locationName.length() > 14) {
            // locationName = locationName.substring(0, 14);
            // }
            // latitude = location.getLatitude();
            // longitude = location.getLongitude();
            // out.println("<p>【位置情報】" + locationName + "</p>");
            // out.print("<p>  緯度:" + latitude + " ");
            // out.println("経度:" + longitude + "</p>");
            // } else {
            // out.println("<p>【位置情報】</p>");
            // out.println("<p>  緯度: 経度: </p>");
            // }

            Comments comments = mediaData.getComments();
            if (comments != null && comments.getComments() != null) {
                System.out.println("### コメント ###");
                for (CommentData commentData : comments.getComments()) {
                    out.println("<p>" + commentData.getText() + "</p>");
                    System.out.println(commentData.getText());
                }
            }

            Likes likes = mediaData.getLikes();
            if (likes != null) {
                int iineCount = likes.getCount();
                System.out.println("### いいね ###");
                System.out.println(iineCount);
                out.println("<p>【いいね】" + iineCount + "</p>");
            }

            out.println("</div>");
            i++;
        }

        out.println("</body>");
        out.println("</html>");
        out.close();

    }

}
```

TorafuguServlet.java

```
package jp.ac.keio.haruyamalabsns;

import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.jinstagram.auth.InstagramAuthService;
import org.jinstagram.auth.oauth.InstagramService;

@WebServlet("/TorafuguServlet")
public class TorafuguServlet extends HttpServlet {

	private static final long serialVersionUID = 1L;

	public static Set<String> sessionSet = new HashSet<String>();

	@Override
	public void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {

		HttpSession httpSession = null;

		try {
			response.setContentType("text/html; charset=UTF-8");

			httpSession = request.getSession(true);

			String hashtag = request.getParameter("hashtag");

			if (hashtag != null) {
				httpSession.setAttribute("hashtag", hashtag);
			}

			InstagramService service = new InstagramAuthService()
					.apiKey(Constants.ACCESS_ID).apiSecret(Constants.ACCESS_SECRET)
					.scope("public_content")
					// .scope("basic public_content follower_list relationships")
					.callback(Constants.SERVER_URL2).build();

			String authorizationUrl = service.getAuthorizationUrl();

			response.sendRedirect(authorizationUrl);

		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public static boolean containsSessionId(String sessionId) {
		return sessionSet.contains(sessionId);
	}

	public static void removeSessionId(String sessionId) {
		sessionSet.remove(sessionId);
	}

}
```

Constants.java

```
package jp.ac.keio.haruyamalabsns;

public class Constants {

	public static final String HOST = "xx.xx.xx.xx";

	public static final String SERVER_URL1 = "http://" + HOST
			+ ":8080/Torafugu/TorafuguServlet";

	public static final String SERVER_URL2 = "http://" + HOST
			+ ":8080/Torafugu/TorafuguService";

	public static final String CLIENT_ID = "xxx";

	public static final String CLIENT_ID2 = "xxx ";

	public static final String CLIENT_ID3 = "xxx ";

	public static final String CLIENT_ID4 = "xxx ";

	/* HaruyamaLab01 */
	public static final String CLIENT_ID5 = "xxx ";

	public static final String CLIENT_SECRET = "xxx";

	public static final String CLIENT_SECRET2 = "xxx";

	public static final String CLIENT_SECRET3 = "xxx";

	public static final String CLIENT_SECRET4 = "xxx";

	/* HaruyamaLab01 */
	public static final String CLIENT_SECRET5 = "xxx";

	public static final String ACCESS_ID = CLIENT_ID5;

	public static final String ACCESS_SECRET = CLIENT_SECRET5;

}
```

(2) ServerApi プロジェクト

プロジェクト構成例

```
root フォルダ
 |    beacon.properties
 |
 ├──lib
 |        postgresql-42.1.1.jar
 |
 └──src
     └──jp
         └──ac
             └──keio
                 └──haruyamalabsns
                     └──serverApi
                             ApiManager.java
                             CommandExecutor.java
                             DbManager.java
                             ServerException.java
                             Utility.java
```

ApiManager.java

```
package jp.ac.keio.haruyamalabsns.serverApi;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

public class ApiManager {

	private static Properties properties = new Properties();

	public static void main(String[] args) {
		ApiManager apiManager = new ApiManager();
		try {
			if (apiManager.existsProcess()) {
				System.out.println("既に ApiManager プロセスが起動しているため，本プロセスを終了します。");
				System.exit(0);
			}
			apiManager.initialize();

			String targetRootFolderPath = properties
					.getProperty("TARGET_ROOT_FOLDER_PATH");
			if (targetRootFolderPath == null) {
				targetRootFolderPath = "D:¥¥inetpub¥¥ftproot";
			}

			System.out.println(Utility.getNowTime()
					+ " ############ アップロード開始 ############");
			apiManager.uploadFiles(targetRootFolderPath);
			System.out.println(Utility.getNowTime()
					+ " ############ アップロード終了 ############");

		} catch (UnsupportedEncodingException e) {
			System.out.println(Utility.printContents(e));
			System.exit(-1);
		} catch (FileNotFoundException e) {
			System.out.println(Utility.printContents(e));
			System.exit(-2);
		} catch (IOException e) {
			System.out.println(Utility.printContents(e));
			System.exit(-3);
		} catch (ServerException e) {
			System.out.println(Utility.printContents(e));
			System.exit(-4);
		} catch (InterruptedException e) {
			System.out.println(Utility.printContents(e));
			System.exit(-5);
		}
	}

	public boolean existsProcess() throws InterruptedException, IOException {

		ProcessBuilder pb = new ProcessBuilder("jps");
		Process process = pb.start();
		process.waitFor();

		InputStream is = process.getInputStream();

		int processNum = 0;
		BufferedReader br = null;
```

```
        try {
            br = new BufferedReader(new InputStreamReader(is));
            String line;
            while ((line = br.readLine()) != null) {
                if (line.indexOf("ApiManager") != -1) {
                    processNum++;
                }
            }
        } finally {
            if (br != null) {
                br.close();
            }
        }

        if (processNum > 1) {
            return true;
        }

        return false;
    }

    public void initialize() throws UnsupportedEncodingException, FileNotFoundException,
            IOException {
        String filePath = "beacon.properties";
        properties.load(new InputStreamReader(new FileInputStream(filePath), "UTF-8"));
    }

    public void uploadFiles(String rootDirPath) throws ServerException, IOException,
            InterruptedException {

        File file = new File(rootDirPath);
        if (!file.isDirectory()) {
            throw new ServerException("パス不正。フォルダパスを指定して下さい。:" + rootDirPath);
        }

        for (String filePath : file.list()) {
            File subfile = new File(rootDirPath, filePath);
            if (!subfile.isDirectory()) {
                // ディレクトリでない場合は飛ばす
                continue;
            }

            // ディレクトリ配下のファイルをアップロードする
            uploadPics(subfile);
        }

    }

    private void uploadPics(File targetDir) throws IOException, InterruptedException,
            ServerException {

        final String FORMAT_HASHTAG = "#%s";

        final String FORMAT_COMMENT_FILE = "%s.txt";

        final String FORMAT_JPG_FILE = "%s¥¥%s.jpg";

        String uwscCommandPath = properties.getProperty("UWSC_COMMAND_PATH");
        if (uwscCommandPath == null) {
            uwscCommandPath = "D:¥¥tool¥¥uwsc523¥¥UWSC.exe";
        }

        String uploadInstagramUwscPath = properties
                .getProperty("UPLOAD_INSTAGRAM_UWSC_PATH");
        if (uploadInstagramUwscPath == null) {
            uploadInstagramUwscPath = "D:¥¥instagram¥¥UploadInstagram.uws";
        }

        String removeFolderPath = properties.getProperty("REMOVE_FOLDER_PATH");
        if (removeFolderPath == null) {
```

```
            removeFolderPath = "D:¥¥instagram¥¥remove";
        }

        String uploadFilePath;

        String hashtagValue = targetDir.getName();

        String hashtag = String.format(FORMAT_HASHTAG, hashtagValue);

        String comment;

        String targetDirPath = targetDir.getAbsolutePath();

        List<String> uplaodFileList = getUploadFileList(targetDir, targetDirPath);
        if (uplaodFileList.isEmpty()) {
            // アップロードファイルがない場合は終了
            return;
        }

        for (String uploadFileName : uplaodFileList) {

            // コメントファイルの参照
            String commentFileName = String.format(FORMAT_COMMENT_FILE, uploadFileName);
            File commentFile = new File(targetDirPath, commentFileName);
            comment = getComment(commentFile);

            uploadFilePath = String
                    .format(FORMAT_JPG_FILE, targetDirPath, uploadFileName);

            System.out.println(Utility.getNowTime() + " " + uploadFilePath + ","
                    + hashtag);

            CommandExecutor executor = new CommandExecutor(uwscCommandPath,
                    uploadInstagramUwscPath, uploadFilePath, hashtag, comment);

            // UWSC を実行し，Instagram にファイルをハッシュタグとコメント付きでアップロード
            if (!executor.execute()) {
                throw new ServerException("  アップロードに失敗しました。");
            }

            DbManager dbManager = DbManager.getInstance();
            try {
                dbManager.openConnection();
                dbManager.insertHashtag(hashtagValue);
                dbManager.closeConnection();
            } catch (ClassNotFoundException e) {
                // DB の例外は無視
                System.out.println(Utility.printContents(e));
            } catch (SQLException e) {
                // DB の例外は無視
                System.out.println(Utility.printContents(e));
            }

            File removeDir = new File(removeFolderPath, targetDir.getName());
            String removeDirRoot = removeDir.getAbsolutePath();
            if (!removeDir.exists()) {
                if (!removeDir.mkdirs()) {
                    throw new ServerException("アップロードファイルの移行先フォルダを作成できません。:"
                            + removeDirRoot);
                }
            }

            // アップロードファイルを削除用フォルダに移動
            File uploadFile = new File(uploadFilePath);
            File removeUploadFile = new File(removeDir, uploadFile.getName());
            if (!uploadFile.renameTo(removeUploadFile)) {
                // SNS にアップロード済みのため，移行できない場合は削除
                uploadFile.delete();
                System.out.println("アップロードファイルを移行先できません。:"
```

```
                        + removeUploadFile.getAbsolutePath());
            }

            // コメントファイルを削除用フォルダに移動
            if (commentFile.exists()) {
                File removeCommentFile = new File(removeDir, commentFile.getName());
                if (!commentFile.renameTo(removeCommentFile)) {
                    // SNS にアップロード済みのため，移行できない場合は削除
                    commentFile.delete();
                    System.out.println("コメントファイルを移行先できません。:"
                            + removeCommentFile.getAbsolutePath());
                }
            }
        }

    }

    private String getComment(File commentFile) throws IOException {
        StringBuilder builder = new StringBuilder();
        if (!commentFile.exists()) {
            return builder.toString();
        }

        BufferedReader br = null;
        try {
            br = new BufferedReader(new InputStreamReader(
                    new FileInputStream(commentFile), "UTF-8"));
            String line;
            while ((line = br.readLine()) != null) {
                builder.append(line + "¥n");
            }
        } finally {
            if (br != null) {
                br.close();
            }
        }

        return builder.toString();
    }

    private List<String> getUploadFileList(File targetDir, String targetDirPath) {

        List<String> uplaodFileList = new ArrayList<String>();

        for (String fileName : targetDir.list()) {
            File file = new File(targetDirPath, fileName);

            if (!file.isFile() || !file.getPath().endsWith(".jpg")) {
                // ファイルでない場合は飛ばす。jpg ファイル以外も飛ばす。
                continue;
            }

            int lastPosition = fileName.lastIndexOf('.');
            uplaodFileList.add(fileName.substring(0, lastPosition));
        }

        return uplaodFileList;
    }

}
```

CommandExecutor.java

```
package jp.ac.keio.haruyamalabsns.serverApi;

import java.io.IOException;
import java.io.InputStream;

public class CommandExecutor {

	private ProcessBuilder pb;

	public CommandExecutor(String... commandLines) {
		pb = new ProcessBuilder(commandLines);
	}

	public boolean execute() throws IOException, InterruptedException {
		boolean result = false;
		pb.redirectErrorStream(true); // 標準エラーを標準出力にマージする
		Process process = pb.start();
		InputStream is = process.getInputStream();
		try {
			while (is.read() >= 0)
				; // 標準出力だけ読み込む
		} finally {
			is.close();
		}

		if (process.exitValue() == 100) {
			//UploadInstagram.uws の正常終了コードは 100 を返す
			result = true;
		}
		return result;
	}

}
```

DbManager.java

```
package jp.ac.keio.haruyamalabsns.serverApi;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;

public class DbManager {

	// localhost
	public static final String HOST = "localhost";

	public static final String URL = "jdbc:postgresql://" + HOST + ":5432/HaruyamaLab";

	public static final String USER = "HaruyamaLab";

	public static final String PASSWORD = "HaruyamaLab";

	public static final String INSERT_HASHTAG_FORMAT = "INSERT INTO HASHTAG (hashtag) VALUES (?)";

	public static final String SELECT_COUNT_HASHTAG_FORMAT = "SELECT COUNT(*) AS cnthash FROM
HASHTAG WHERE hashtag=?";

	public static DbManager dbManager = new DbManager();

	private Connection conn = null;

	private DbManager() {

	}
```

```
    public static DbManager getInstance() {
        return dbManager;
    }

    public void openConnection() throws ClassNotFoundException, SQLException {
        if (conn != null) {
            return;
        }

        Class.forName("org.postgresql.Driver");
        Properties props = new Properties();
        props.setProperty("user", USER);
        props.setProperty("password", PASSWORD);
        props.setProperty("ssl", "false");
        conn = DriverManager.getConnection(URL, props);
    }

    public void insertHashtag(String hashtag) throws SQLException {

        conn.setAutoCommit(false);
        PreparedStatement ps = conn.prepareStatement(SELECT_COUNT_HASHTAG_FORMAT);
        ps.setString(1, hashtag);
        ResultSet rs = ps.executeQuery();
        rs.next();
        int num = rs.getInt("cnthash");
        if (num != 0) {
            return;
        }

        ps = conn.prepareStatement(INSERT_HASHTAG_FORMAT);
        ps.setString(1, hashtag);
        num = ps.executeUpdate();
        conn.commit();
        System.out.println("insertHashtag" + hashtag);

    }

    public void closeConnection() throws SQLException {
        if (conn != null) {
            conn.close();
            conn = null;
        }
    }

}
```

ServerException.java

```
package jp.ac.keio.haruyamalabsns.serverApi;

public class ServerException extends Exception {

    public ServerException(String message) {
        super(message);
    }

}
```

Utility.java

```
package jp.ac.keio.haruyamalabsns.serverApi;

import java.io.PrintWriter;
import java.io.StringWriter;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Utility {

    public static String getNowTime() {
        Date now = new Date();
```

```
        SimpleDateFormat dateFomat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
        return dateFomat.format(now);
    }

    public static String printContents(Exception e) {
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);
        e.printStackTrace(printWriter);
        return stringWriter.toString();
    }
}
```

(3) UWSC 実行環境

フォルダ構成例

```
root フォルダ
 │   beacon.properties
 │   initdb.bat
 │   postgresql.bat
 │   serverapi.jar
 │   ServerApi.log
 │   UploadFile.bat
 │   UploadInstagram.uws
 │
 └──ddl
        initdb.sql
```

initdb.bat

```
set PGPASSWORD=HaruyamaLab

"D:¥Program Files¥PostgreSQL¥9.5¥bin¥psql.exe" -f ddl¥initdb.sql -U HaruyamaLab
```

UploadFile.bat

```
SET CURRENT_PATH=%~dp0
SET SERVER_LIB=%CURRENT_PATH%¥serverapi.jar
SET LOG_FILE=%CURRENT_PATH%¥ServerApi.log

java -cp %SERVER_LIB% jp.ac.keio.haruyamalabsns.serverApi.ApiManager >> %LOG_FILE%
```

UploadInstagram.uws

```
//頻繁に Gramblr のバージョンが更新されるので注意。
GRAMBLR = "Gramblr64 v2.7.5"
ID = GETID(GRAMBLR,"Chrome_WidgetWin_1")

IFB !STATUS(ID, ST_ACTIVE) //  ウィンドウがアクティブかチェック
  CTRLWIN(ID, ACTIVATE) //  アクティブにする
ENDIF

//  位置調整（左上）
ACW(ID,0,0,1)

//Upload Now!クリック
BTN(LEFT,CLICK,138,311,2000)

//待ち
SLEEP(2)

//  ファイル選択エリアクリック
BTN(LEFT,CLICK,768,536,2000)

//待ち
SLEEP(2)
```

```
FileDialogID=GetID(GET_ACTIVE_WIN)

SENDSTR(FileDialogID,PARAM_STR[0])

Return = CLKITEM(FileDialogID, "開く",CLK_LEFTCLK)

//待ち
SLEEP(2)

//Save ボタンクリック
BTN(LEFT,CLICK,1075,334,2000)

//待ち
SLEEP(2)

//Continue ボタンクリック
BTN(LEFT,CLICK,962,543,2000)

//待ち
SLEEP(2)

//Caption エリアをクリック
BTN(LEFT,CLICK,962,543,2000)

UploadDetailID = GETID(GRAMBLR,"Chrome_WidgetWin_1")

//待ち
SLEEP(2)

//SENDSTR で直接テキストエリアに入力できないため下記で対応

// グリップボードにハッシュタグをコピー
SENDSTR(0,PARAM_STR[1] + "<#CR>" + PARAM_STR[2])

//Caption:エリアにハッシュタグを貼り付け
SCKEY(UploadDetailID,VK_CTRL,V)

//Send!ボタンクリック
BTN(LEFT,CLICK,909,848,2000)

//待ち
SLEEP(2)

//アップロード待ち
FOR A = 10 TO 0 STEP -1
FUKIDASI("後" + A + "秒",100,100,0,40)
SLEEP(1)
NEXT

//終了コードを 100 とする
EXITEXIT 100
```

initdb.sql

```
DROP INDEX IF EXISTS HASHTAG_INDEX;
DROP INDEX IF EXISTS BEACON_INDEX;
DROP TABLE IF EXISTS HASHTAG;
DROP TABLE IF EXISTS BEACON;

CREATE TABLE HASHTAG(id SERIAL PRIMARY KEY,hashtag varchar(256));
CREATE TABLE BEACON(id SERIAL PRIMARY KEY,uuid varchar(128),hashtag varchar(256));
CREATE INDEX HASHTAG_INDEX ON HASHTAG(id);
CREATE INDEX BEACON_INDEX ON BEACON(id);
```

11.2. アンケート兼インタビュー

以下の項目に対して選択もしくはコメントの記入をお願いします。

(1) 年齢(選択)
- 20 代
- 30 代
- 40 代
- 50 代
- 60 以上

(2) 性別(選択)
男・女

(3) Instagram などの SNS を使用したことがありますか？(選択)
はい・いいえ

(4) (3)で「はい」を選択した方に質問します。SNS に写真投稿する時に，もっと楽に投稿できると良いと思えますか？例えば，写真を撮ってコメント入力したらそのまま投稿されるぐらいの容易さ感。(選択)
はい・いいえ

(5) (3)で「はい」を選択した方に質問します。位置情報を使用したことがありますか？(選択)
はい・いいえ

(6) (4)で「はいを」選択した方に質問します。位置情報を使用する際にもっと楽に投稿できると良いと思えますか？例えば，自分で位置情報を入力または位置スポットを選択するような操作を省略できるなど。(選択)
はい・いいえ

(7) 利用されている SNS と比較して，本システムで良かった点があればご記入下さい。

(8) 本システムで改善点があればご記入下さい。