卒業論文

デジタル版しおりアプリの開発

関西学院大学理工学部

情報工学課程 西谷研究室

37022463 山本果音

2026年3月

概 要

旅行アプリ「旅しお」では旅行計画を簡単に立てることが可能であるが, 予定作成時の情報整理や視覚的な把握が不十分であり, 継続的な利用が困難である. 本研究では, 旅行前の計画段階において, ユーザーがスムーズに情報を整理・共有できることを目的とし, デジタル版しおりの開発を行った. 具体的な特徴として, 旅行名や説明文をキーとした検索機能, 旅行日付に基づく時系列整理, 目的地の地図表示機能, 移動時間表示機能, 訪問順序入れ替え機能, チェックリスト機能を実装し, 新たな予定を立てやすくした. 結果としてユーザーは旅行計画を効率的に立てることができ, 継続的な利用が可能になった.

目 次

図 目 次

第1章　序論

修学旅行などのイベントでは, 参加者に対して事前にしおりが配布されることが多い. しおりは旅程の把握や所持品の確認, 集合時間の共有などを目的とした重要な情報媒体であり, 旅行の準備や当日の行動において欠かせない役割を果たしている. 近年では, スマートフォンの普及により, 紙媒体に代わって Web サービスを活用した旅行情報の閲覧・共有のニーズが高まっており,「旅しお」[1] のような Web アプリも登場している. しかし, これらのサービスにはいくつかの課題が存在する.

1 つ目は, 旅行履歴の表示が作成日や更新日を基準としており, 実際の旅行日付に基づいた時系列での整理がされていない点である. そのため, 過去の旅行の実施状況を視覚的に把握することが困難である.

2 つ目は, 旅行名や説明文に対するキーワード検索機能が存在しないため, ユーザーはスクロール操作によって目的のしおりを探す必要がある点である.

3 つ目は, 地図表示機能がないため, 目的地の位置関係や全体像を視覚的に理解することが難しく, 利用者は外部ツールや手作業で補完しなければならない.

これらの課題は, 旅行計画の際にユーザーにとって大きな障壁となっており, 情報の整理や視覚的な理解の低下が考えられる. そこで本研究では, 視覚的な理解と旅行後の振り返りを行える「デジタル版しおり」として機能する旅行アプリの開発を目指す.

第2章　手法

2.1　開発手法

本研究では, 限られた開発期間内で効率的に成果を上げるために, 機能ごとに優先順位を設定し, それぞれについて計画・設計・実装・テストの工程を小刻みに繰り返す手法を採用した. __各段階で得られた知見や, 研究室内のメンバーからのフィードバックをもとに, 仕様や UI の見直しを行いながら, 柔軟に改善を重ねた. これにより, 利用者の視点を反映した機能の追加や調整が可能となり, アプリケーションの完成度と実用性を段階的に高めることができた.

2.1.1　Dango

開発環境として,Python で実装されたサーバーサイド Web アプリケーションフレームワークである Django を選定した [2]. Django を選定した理由は以下の 3 つである.

1 つ目は, ユーザー認証や管理画面, データベース操作などの機能が初期状態で組み込まれており, 追加設定を行わずとも迅速に開発を開始できる点である. これにより, 開発初期の環境構築にかかる手間を大幅に削減することが可能となった.

2 つ目は, データベース操作をすべて Python で記述できるため, 学習コストが低い点である. Django では, モデルの定義からマイグレーションの実行までを一貫して Python で記述できるため,SQL の詳細な知識がなくても複雑なデータ構造の設計や変更が容易に行える.

3 つ目は, セキュリティ対策がフレームワークに標準で備わっている点である. Django はクロスサイトスクリプティング (XSS) や SQL インジェクションなどの脆弱性に対して, 初期状態で対策が施されている. そのため, 開発者が個別にセキュリティ機能を実装する必要がなく, 安全性の高い Web アプリケーションを効率的に構築することができた.

また, 実装には Python, HTML, JavaScript, CSS を使用し,UI ライブラリとしては Bootstrap を採用した．開発にあたっては Copilot[3] を参照しながら, 開発を行った.

2.1.2 Heroku

第3章　結果と考察

3.1　アプリ機能

今回，開発したアプリでは大きく分けて 4 つの機能を実装した．それぞれの機能は図 3.1 の通りである．

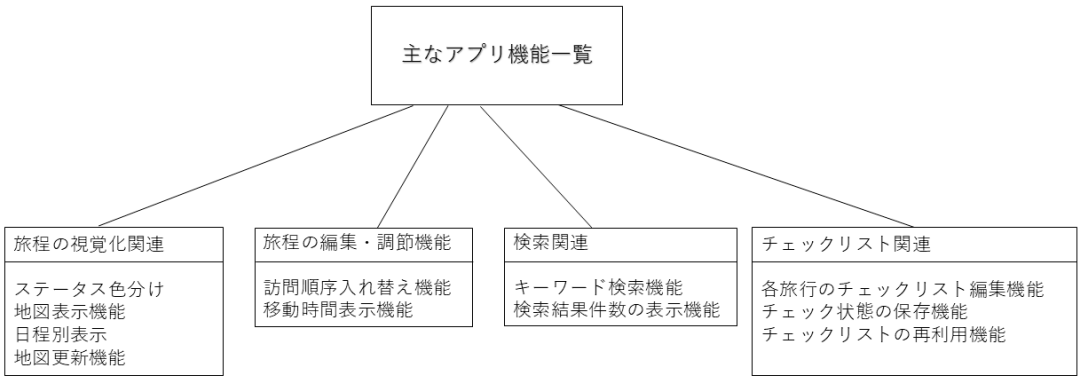

図 3.1: アプリ機能一覧.

3.1.1　旅程の視覚化管理

ユーザーが旅行の全体像や日程ごとの計画を直感的に把握できるよう, 複数の視覚化機能を実装した.

まず, 登録された旅行期間に基づいて, 各旅行を「予定」「旅行中」「終了」「未定」の 4 つのステータスに分類し, 色分けによって一覧画面上に表示する機能を実装した．これに

より, ユーザーは「どこに行ったか」「これからどこに行くか」といった旅行の進行状況を一目で把握できるようになった.

図 3.2: 旅行一覧画面.

また, 旅行詳細画面では,Google Maps API を用いて登録された観光場所を地図上に表示する機能を導入した. これにより, 各観光地の位置関係や移動ルートを視覚的に確認でき, 旅行計画全体の構造を把握しやすくなった.

さらに, 旅行期間を事前に入力しておくことで, 旅行の日数分だけタブを動的に生成する機能を実装した. 旅行に含まれる観光場所を「全体」「1 日目」「2 日目」など日程ごとに切り替えて表示する機能も併せて導入し, ユーザーはタブを操作することで, 旅行全体の流れを把握しながら, 各日の予定を個別に確認・編集することが可能となった. これにより, 日程ごとの計画を視覚的に整理しやすくなり, 過不足のあるスケジュールや移動の偏りにも気づきやすくなった. また, 編集対象の日付を明確に切り替えられることで, 誤操作の防止にもつながっている.

タブの切り替えに連動して, 地図上に表示される観光地もその日に対応するもののみに自動で更新されるため, 日程ごとの旅程がより視覚的に管理しやすい設計となっている.

これらの機能により, ユーザーは旅行の計画段階において, 旅程を効率的に把握・調整できるようになり、直感的な旅行プランニングが可能となった.

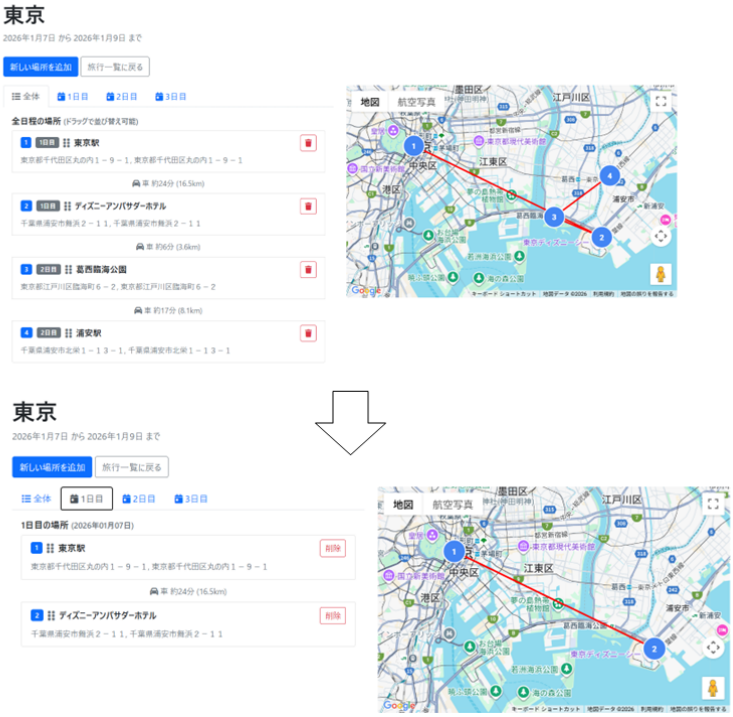

図 3.3: 全体から 1 日目のタブを押したときの変化.

3.1.2 旅程の編集・調節機能

旅行の計画を柔軟に調整できるよう, 旅程編集に関する 2 つの機能を実装した.

まず, 観光場所の訪問順序をドラッグすることで自由に入れ替える機能を導入した. これにより, ユーザーは移動効率や興味関心に応じて旅程を柔軟に再構成できるようになった. 従来のように順序を変更するたびに項目を削除し, 再度追加し直す必要がなくなったことで, 操作性が向上し, 旅程編集の手間が大幅に軽減された.

また, 各観光地間の移動時間を自動で算出・表示する機能を実装した. 算出されるのは車での移動を想定した所要時間であり, ユーザーは移動にかかる時間を考慮しながら, 現実的かつ無理のないスケジュールを立てることが可能となった. さらに, 移動時間が可視化されたことで, 訪問地の組み合わせや順序の検討がより直感的かつ効率的に行えるようになった.

これらの機能は, 旅行の編集・調節を支援する上で不可欠な要素であり, ユーザーが旅程全体の流れを把握しながら, 日程ごとの詳細な計画を直感的に調整できる環境を提供している.

図 3.4: 移動時間表示画面.

3.1.3 検索機能

旅行情報の登録・閲覧・編集を効率的かつ直感的に行えるよう,2 つの検索機能を実装した.

旅行一覧画面に多くの旅行が表示されている場合, 目的の旅行を探し出すのに時間がかかることがある. そこで, 旅行名や説明欄に含まれる単語をもとに部分一致によるキーワード検索を行い, 該当するデータを絞り込めるようにした. いずれの検索においても, 大文字・小文字を区別しない部分一致検索を実現しているため, ユーザーは入力時に文字の大小を意識する必要がない. そのため, 検索漏れを防ぐことができ, よりスムーズに目的の情報へアクセスできるようになった. また, 検索結果の件数を取得し,「〇件」という形式で表示することで, 検索結果の量を一目で把握できるようにした.

さらに, 施設名を入力するだけで, 対応する位置に地図上のピンを自動で立てられる機能を導入した. これにより, ユーザーは住所を調べて緯度経度を入力する手間なく, 直感的に観光地を登録できるようになった.

3.1.4 チェックリスト機能

旅行準備を効率的に進められるよう, チェックリストに関する複数の機能を実装した. まず, 旅行ごとに標準のチェックリスト設定を適用できる機能を導入した. これにより, ユーザーは毎回ゼロからリストを作成する必要がなく, あらかじめ用意された基本項目をもと

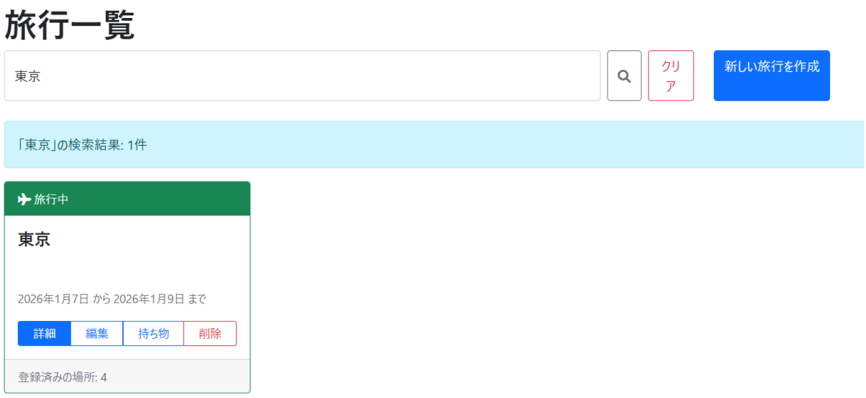

図 3.5: 検索結果画面.

図 3.6: 施設名の入力画面.

に効率よく準備を開始できるようになった．さらに, 基本項目をテンプレート化することで,「今回だけ書き忘れた」といった漏れを防ぐ効果も期待できる．

図 3.7: チェックリスト設定画面.

また, 各旅行ごとに持ち物や準備項目を自由に編集できるチェックリスト編集機能を導入した．これにより, ユーザーは旅行の目的や季節, 同行者に応じて, 必要な項目を柔軟に追加・削除することが可能となった．状況に応じたカスタマイズが行えることで, より実際のニーズに即した準備ができるようになり, 忘れ物や準備漏れの防止にもつながる．

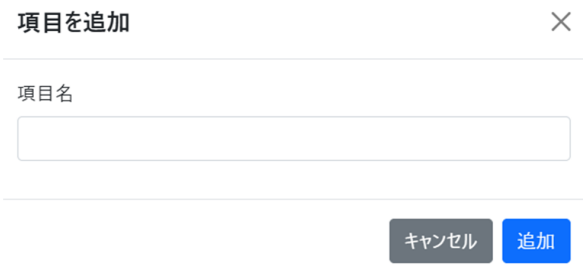

図 3.8: チェックリスト追加画面.

さらに, 各項目のチェック状況を保持する機能を実装し, ユーザーが確認済みの持ち物を記録できるようにした．これにより, 準備の進捗状況を可視化しながら, 持ち物の確認状況を一目で把握できるようになり, 確認漏れや二重確認の手間を軽減することが可能となった．チェックの状態は旅行ごとに保存されるため, 複数の旅行を並行して準備する場合で

も, 各旅程の進捗を個別に管理できる. これらの機能は, 旅行前の準備を支援する上で重要

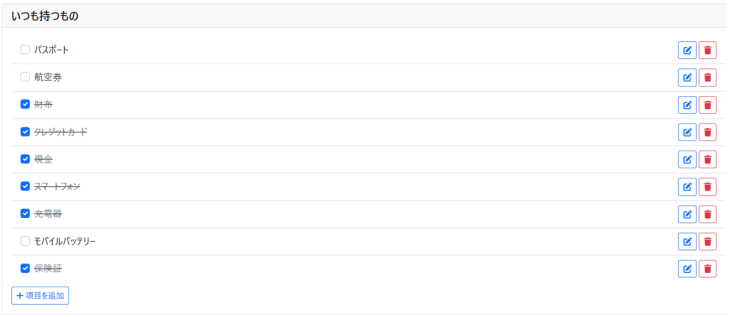

図 3.9: チェックリストの管理画面.

な役割を果たしており, ユーザーが安心して出発できるよう, 実用性と柔軟性を兼ね備えたチェックリスト管理環境を提供している.

第4章　まとめ

本研究では，Ruby で書かれたサーバー側 Web アプリケーションフレームワークである Rails を開発環境として，データを共有・見返す作業を癖付けることを目的としたグループウェアアプリ Habit man を開発した．

開発手法として事前に全ての機能やサービスの詳細な要件，作業スケジュールを立てるウォーターフォール開発ではなく，優先順位を付けて重要な機能やサービスを段階に分けて開発し，ユーザーから得た反応から方向性や仮説を見出し，ユーザーへ価値を素早く届け，実戦投入の学びから素早く改善を行うというサイクルを確立するアジャイル開発をした．結果として，研究室のメンバーから早い段階でフィードバックを頂けたのでより良いグループウェアアプリへと修正することができた．また，機能的な側面としてカレンダー表示やタグ付け機能などの複雑な実装はパッケージマネージャーである gem に配布されているライブラリを用いて実現させた．

日付を第 1 キーとしてデータを保存することによって，日付ごとのデータ管理が容易になり，データの検索が高速化した．加えて，データをただ単に溜めていくだけではなく，独自のアルゴリズムから不必要と判定されたデータをアーカイブに移動することによってデータベースやストレージの使用量を最適化できた．

結果として，グループメンバーのデータ共有・見返す作業の習慣化に繋げることができた．

謝辞

本研究を進めるにあたり, 御指導いただきました西谷滋人教授には心から感謝いたします. また, 実際にアプリを使用していただき, 貴重な意見を下さった西谷研究室に所属している皆さまにも大変お世話になりました. お礼申し上げます.

参考文献

[1] Discord - https://discord.com (accessd on 21 Nov 2023).

[2] 野口 悠紀雄, 「超」整理法—情報検索と発想の新システム (中公新書), 中央公論新社, (1993).

[3] Slack - https://slack.com/intl/ja-jp/help/articles/115004071768-Slack-%E3%81%A8%E3%81%AF (accessd on 6 Jan 2024).

[4] Agile software development - https://en.wikipedia.org/wiki/Agile_software_development (accessd on 20 Nov 2023).

[5] Ruby on Rails - https://rubyonrails.org/ (accessd on 20 Nov 2023).

[6] まつもとゆきひろ - https://ja.wikipedia.org/wiki/%E3%81%BE%E3%81%A4%E3%82%82%E3%81%A8%E3%82% (accessd on 18 Dec 2023).

[7] Qiita - https://help.qiita.com/ja/articles/qiita (accessd on 18 Dec 2023).

[8] RubyGems - https://rubygems.org/ (accessd on 18 Dec 2023).

[9] Heroku - https://www.heroku.com/what (accessd on 18 Dec 2023).

[10] Heroku Add-ons - https://elements.heroku.com/addons (accessd on 18 Dec 2023).

[11] Convention over Configuration - https://en.wikipedia.org/wiki/Convention_over_configuration (accessd on 18 Dec 2023).

[12] Habit man - https://membermanagementapp-d0c147b97826.herokuapp.com/

[13] Heroku Postgres - https://devcenter.heroku.com/ja/articles/heroku-postgresql (accessd on 29 Jan 2024).

[14] Active Record の基礎 - https://railsguides.jp/active_record_basics.html (accessd on 29 Jan 2024).

[15] Rails コア開発環境の構築方法 - https://railsguides.jp/development_dependencies_install.html (accessd on 29 Jan 2024).

[16] Heroku Scheduler - https://devcenter.heroku.com/ja/articles/scheduler (accessd on 29 Jan 2024).

[17] Strong parameters - https://railsguides.jp/action_controller_overview.html#strong-parameter (accessd on 3 Dec 2023).