大規模言語モデルを用いたアプリ開発支援手法

黒田 慧[1]　高田 秀志[1]

概要：日本では，小学校，中学校でプログラミング教育が必修化されるなどプログラミング教育への注目が高まっている．しかし，プログラミングは専門性が高く，特にアプリケーションの実装は初学者にとってはハードルが高い．その原因には，複数の技術を習得する必要があることや，実装前に機能や要件を細分化し，完成までのプロセスを明確化する必要があることなどが挙げられる．そこで本研究では，生成 AI を用いて個人の知識や経験に基づいた開発手順の提示を行うことで，効率的に学習を行うことができるシステムを提案する．

1. はじめに

日本のプログラミング教育は，小学校では 2020 年度から，中学校では 2021 年度から必修化され，論理的思考力や問題解決能力の育成が重視されている．高校においても段階的に拡充が進められ，実践的な ICT 活用力の育成が図られている．社会的には，デジタル人材不足を背景にその重要性が一層高まっており，民間企業やオンライン教材を通じた学習機会も増加している．

しかし，プログラミング学習は専門性が高く，初学者にとって依然としてハードルが高い．特にアプリケーションを開発するには，複数のプログラミング言語やデータベースの知識，設計ノウハウ，ディレクトリ構成の理解など，多岐にわたる要素が求められる．さらに，それらを一通り学習しても，実践経験が少なければ知識をどう組み合わせればよいのか分からず，何から手を付けるべきか迷うことが多い．また，機能や要件を明確化し，完成までのプロセスを具体化することも重要であるが，これには知識と経験が不可欠であり，初学者には困難な場合が少なくない．プロセスがうまく描けなければ，実装に過度な時間を費やしたり，途中で挫折したりする問題が生じる．

本研究では，このような課題を解消するため，ユーザの技術経験や開発対象に応じて，開発の各工程で必要な作業内容を提示するシステムを提案する．本システムは生成 AI を活用し，ユーザの経験や知識に基づいた適切な情報を提供することで効率的な支援を行う．さらに，ユーザが希望する技術やアプリケーションの内容に合わせて，適切な支援を実現することを目指す．

[1] 立命館大学情報理工学部

2. 関連研究

村山ら [1] は，初学者むけのアプリ開発支援の研究を行った．この研究では，プログラミング未経験者でも iPhone と Android の両方に対応したアプリケーションを簡単に開発できるシステムを提案している．特に，GUI を活用してグリッド線に沿ったウィジェット配置が可能な点が特徴であり，iPhone と Android でのアプリケーション開発の敷居を下げることを目標とした．実験を通して，初心者でも短時間でアプリを開発できることが確認された．我々の研究と比較して，初学者のアプリケーション開発を支援するという点では共通している．一方で，この研究では GUI を活用してウィジェット配置を行い，最終的にはシステムを用いて自動的に各プラットフォームのプログラムに変換していたのに対し，本研究では，生成 AI が生成した手順を参照してユーザ自身が実際に開発を行う．

Mohamad Iyad Al-Khiami ら [2] は，初心者開発者が ChatGPT を活用して Android アプリのコードを生成する手法を調査した．この研究は，プログラミングの初学者が，チャットボットとの対話を通じてコード作成やデバッグを行い，アプリ開発が成功するプロセスを検討した．結果として，ChatGPT は効率性と生産性の向上に寄与し，開発初心者が複雑な問題に取り組む際に有用であることが分かった．我々の研究において，生成 AI を用いて初学者のアプリケーション開発を支援するという点では共通している．一方で，この研究ではユーザ自身がアプリに必要とされる機能を細分化し，その内容を踏まえて生成 AI にコードを生成させたのに対し，本研究では生成 AI を用いて開発手順を生成する．

3. 提案手法

3.1 全体像

本システムでは，以下の流れに沿ってアプリ開発を支援する．

(1) 技術力の推定：ユーザの知識や経験を踏まえた支援を行うため，事前に作成したいくつかの質問にユーザが回答することで技術力を測定する．
(2) チュートリアルの実施：開発に必要な基本技術を習得するため，簡易的な Web アプリケーションの開発を行う．
(3) 開発手順の提示：「技術力の測定」で得た質問の回答を用いて生成 AI が生成した開発手順を提示する．

なお，本研究でユーザがアプリ開発に使用する技術は，フロントエンドを HTML/CSS，バックエンドを Python，Flask，データベースを SQLite とする．また，ユーザがチュートリアルを行った後に開発するアプリとしては以下の 5 つを対象とする．

- Todo アプリ
- カレンダーアプリ
- 天気予報アプリ
- メモアプリ
- 書籍検索アプリ

3.2 技術力の測定

技術力の測定は，事前に作成した 11 問の質問にユーザが回答することで行う．これらの質問は使用する技術に関する基本的な知識を対象としており，回答形式は「はい」「いいえ」の選択式である．

使用する質問を表 1 に示す．なお，質問における「理解している」とは，正しいソースコードを参照した際にその処理内容を把握できる状態を指す．

3.3 チュートリアルの実施

ユーザは，アプリケーションの実装前に生成 AI が事前に生成したチュートリアルを実施する．本研究では対象ユーザとして Flask の使用経験が乏しいことを想定しているため，Flask を用いた簡易的な Web アプリケーションの開発を行う．チュートリアルは，以下の流れで構成されている．

(1) ディレクトリの構成に関する説明
(2) Flask アプリケーションの基本設定とルーティングの実装
(3) 各画面の HTML テンプレートの作成
(4) スタイルシートの作成
(5) アプリケーションの実行

チュートリアルの生成時には，生成 AI へのプロンプトとして，フロントエンドからバックエンドへの通信方法や，Web アプリケーションの基本的なディレクトリ構成が理解できること， 少なくとも 1 回の画面遷移を行うこと等を与えた．

このチュートリアルを通じて，ユーザは Web アプリケーションの基本的な構築手法を習得し，Flask を用いたバックエンド開発やフロントエンドとの通信方法を理解する．また，画面遷移を含むシンプルな機能を実装することで，アプリケーション開発全体の流れを把握し，その後の実践的な開発作業にスムーズに取り組めるようになることを目的としている．

3.4 開発手順の提示

ユーザは実装したいアプリケーションを全体像で述べた 5 つの中から 1 つ選択する．その後，ユーザが選択したアプリ，質問の回答結果から 生成 AI が開発順序を提示する．手順として，「各手順で実装する機能」「その機能を実装するためのステップ」「その機能を実装するにあたってユーザが学習するべき内容」を生成する．生成した手順の例を以下に示す．

(1) プロジェクトのセットアップ
- Flask プロジェクトのフォルダ構成を決める
- 必要なファイルとフォルダを作成する（例: templates, static）
- 仮想環境を構築し、Flask をインストールする
- 学習項目
  - 基本的なファイル操作（ファイルの作成とフォルダ管理）
  - 仮想環境の作成と Python パッケージのインストール

(2) アプリの初期化
- Flask アプリケーションの基本的なセットアップを行う
- アプリの動作確認用に簡単な Python ファイルを作成する
- アプリのルートを設定し、初期ページを表示できるようにする
- 学習項目
  - Python の基本的なプログラムの書き方
  - Flask の基本的なルーティングの設定方法

(3) ホームページの作成
- HTML ファイルを作成し、基本的なレイアウトを定義する
- CSS で基本のスタイルを作成し、HTML に適用する
- ページの表示を確認して修正する
- 学習項目
  - HTML の基本構造（head, body など）
  - CSS の基本的な書き方とスタイル適用方法

(4) データベースの設定

表 1 使用する質問リスト

No.	質問文
1	あなたは Web アプリを開発したことがありますか?
2	あなたは HTML で画面遷移を行う方法を理解していますか?
3	あなたは HTML のフォーム要素 (例: 入力フィールド、ボタン、チェックボックスなど) を使ってデータを送信する方法を知っていますか?
4	あなたは CSS を使って Web ページのレイアウトやスタイルを変更する方法を理解していますか?
5	あなたは CSS フレームワーク (例:Bootstrap) を使用して Web ページをデザインしたことがありますか?
6	あなたは Python でモジュールを使用する方法を理解していますか?
7	あなたは Python でクラスや関数を定義する方法を理解していますか?
8	あなたは Python の条件分岐や繰り返し処理の書き方を理解していますか?
9	あなたは外部の API を使用してデータの取得を行ったことがありますか?
10	あなたはデータベースの正規化の概念を知っていますか?
11	あなたは SQL を使って基本的な CRUD 操作 (作成、読み取り、更新、削除) を実行できますか?

- SQLite データベースのファイルを作成する
- データベース接続を設定し、テーブルを作成する SQL を実行する
- Todo テーブルの作成（カラム: id, task_name, completed）
- 学習項目
  - SQLite データベースファイルの作成方法
  - SQL の基本的な文法とテーブル作成

(5) タスクの追加機能
- HTML フォームを作成し、新しいタスクの入力欄を追加する
- フォームからデータを受け取り、データベースに保存する処理を作成する
- タスク追加後にホームページへリダイレクトする
- 学習項目
  - HTML フォーム要素の使い方（input, button）
  - フォームデータを取得し Python で処理する基本

(6) タスクの表示機能
- データベースからタスクを取得する処理を作成する
- HTML でタスク一覧を表示するためのループ処理を追加する
- 画面でタスク一覧が正しく表示されるか確認する
- 学習項目
  - SQL でデータを取得する方法（SELECT 文）
  - ループ処理でデータを表示する方法

(7) タスクの削除機能
- 各タスクに削除ボタンを追加する
- 削除ボタンが押されたときに該当タスクをデータベースから削除する処理を作成する
- 削除後にホームページへリダイレクトする
- 学習項目
  - SQL でデータを削除する方法（DELETE 文）
  - ボタンクリックでサーバーサイドの処理を呼び出す方法

(8) タスクの完了状態変更機能
- タスクに完了のためのチェックボックスを追加する
- チェックの変更がデータベースに反映されるようにする
- 変更結果が画面に正しく反映されるか確認する
- 学習項目
  - SQL でデータを更新する方法（UPDATE 文）
  - チェックボックスの状態をサーバーに送信する処理

(9) コードの整理とテスト
- 全体のコードを見直し、コメントを追加する
- 各機能が正しく動作するか確認し、バグを修正する
- 不要なコードや無駄な処理を整理して削除する
- 学習項目
  - Python コードにコメントを追加する方法
  - バグを見つけるための基本的なテスト方法

次に，開発手順の生成に用いるプロンプトについて述べる．本システムでは 2 回に分けてリクエストを送ることで開発手順を作成する．1 回目のリクエストで「各手順で実装する機能」のみを生成する．2 回目に「各手順で実装する機能」に対する「その機能を実装するためのステップ」「その機能を実装するにあたってユーザが学習するべき内容」を生成する．

1 回目のリクエスト時に用いたプロンプトには「実装のための開発手順を機能単位で分割し、リスト形式で簡潔に示す」ということを指定し，他の条件として，実装が学習目的であるため，使用する技術について，デプロイやリリースといった工程が不要であること，環境構築を終えていること，実装は必要最低限の機能のみで行い，なるべく簡単に実装できるようにすること，データベースの使用は必須ではないことを指示した．

2 回目のリクエスト時に用いたプロンプトには「技術力の測定で得た回答結果を用いて『その機能を実装するためのステップ』『その機能を実装するにあたってユーザが学習

するべき内容』を提示する」ということを指定し，他の条件として，JavaScript などの他の技術を使用しないこと，必要最低限の機能のみにしてなるべく簡単に実装できるようにすること，ユーザが理解していない部分に関しては開発に関する専門用語を使用しないことを指示し，質問の回答結果を添付した．

3.5 実装

本システムは，Web アプリケーションとして実装されている．フロントエンドにはプログラミング言語に TypeScript，フレームワークに Next.js，Tailwind CSS，ライブラリに React，UI コンポーネントライブラリに Chakra UI を使用している．バックエンドにはプログラミング言語に Python，フレームワークとして Flask，外部の API として OpenAI API を使用した．データベースには MySQL を使用している．

これらの技術を用いて実装した Web アプリケーションを Apache サーバ上に配置する．Apache サーバが Web サーバおよび WSGI モジュールとして機能し，ブラウザからのアクセスを /appdevsupport でフロントエンド，/appdevsupportbe でバックエンドへルーティングする．システムの構成図を図 1 に示す．

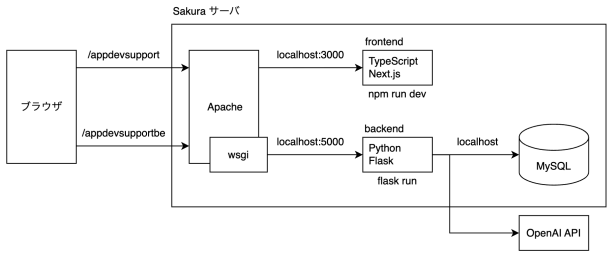

図 1 システム構成図

4. 評価実験

4.1 実験方法

本システムを用いてアプリ開発手順が習得できるようになるかを検証するために，大学生 8 人を対象に評価実験を行った．被験者は，まず，一つのアプリを選択し，その開発手順を作成する．その後，提案システムを用いて先に選択したアプリとは別のものを選択し，実装を行う．最後に，もう一度，これまでに選択したものとは異なるアプリを選択し，その開発手順を作成する．提案システムの効果は提案システムを用いて実装を行う前後で作成した開発手順をスコアリングし，スコアの変化を比較することにより検証する．

「開発手順の作成」では，検索エンジンや生成 AI の使用を禁止し，自力で作成するよう求めた．作成時間は 10 分とした．「提案システムを用いた実装」では，生成 AI の使用を禁止したが，検索エンジンを用いた情報収集は許可した．実装時間は 60 分とした．

4.2 実験結果

本システムを使用して実装を行う前後で作成された開発手順の例を提示する．

実装前に作成された Todo アプリの開発手順を以下に示す．

(1) タスクの管理（登録・閲覧・削除）
- 内容をデータベースで管理

(2) タスク内容の通知

(3) 実行・未実行のチェック

実装後に作成された天気予報アプリの開発手順を以下に示す．

(1) 天気予報の API を取得
- Google など天気関連の API をとってくる

(2) ホーム画面の作成
- 簡単なホーム画面 (天気を表示させるため) を作る

(3) 天気の表示
- 取得した天気情報をホームに表示させる

(4) 追加機能の作成 (検索・日付指定・地域・マップなど)
- あったら便利と思われる追加機能を実装していく

4.3 考察

上に提示した実装前後で作成された開発手順を比較する．実装前に作成された Todo アプリの開発手順では，(1) のタスクの管理（登録・閲覧・削除）のように，複数の機能をまとめたものがあり，適切に細分化できていないものがある．一方で，天気予報アプリの開発手順では，一つの機能や一つの画面に対して手順を作成できていた．また，実装前に作成された開発手順よりも実装後に作成された開発手順の方がそれぞれの手順の詳細を記述することができていることがわかる．

また，どちらの開発手順にも，Todo アプリの開発手順の (2) や天気予報アプリの開発手順の (4) のように余分な機能が含まれていた．しかし，順序に着目すると，Todo アプリでは実装の途中で余分な機能に対する手順があるのに対し，天気予報アプリでは，必要最低限のアプリを実装した後に追加機能の実装を行う手順があるため，実装後に作成された開発手順の方がより一貫した順序になっていると考えられる．

5. おわりに

本研究では，プログラミング初学者にとってアプリケーションの開発はハードルが高く，実装時に何から手を付けるべきか迷う問題や，実装に過度な時間を費やす，あるいは途中で挫折するといった問題に着目した．これらの問題に対し，生成 AI を用いて個人の理解度や経験に基づいた情報を提示することで，効率的に実装を進めることが可能

なシステムを提案した．

今後の課題としては，より個人に適応した情報提示手法の検討や，他の技術およびアプリケーションへの拡張が挙げられる．

参考文献

[1] 村山優弥, 佐藤輝久, 瀬黒雄作, 平野竜希, 平松貴宏, 三沢拓也, 紫合治：iPhone と Android アプリの共通開発支援ツール, 情報処理学会第 75 回全国大会 (2013).

[2] Al-Khiami, M. I. and Jaeger, M.: Leveraging ChatGPT in Android App Development: A Case Study on Supporting Novice Developers in Creating Successful Apps, *Preprints.org* (2023), doi: 10.20944/preprints202307.0660.v1.