Plutos – Streamlining Expense Tracking and Budgeting

Team 10: Payton Chan, Eric Chen, Fondson Lu, Jason Tan, Angela Wang
McMaster University | Department of Computing and Software

Motivation

Young adults often face challenges in managing their finances effectively, especially when it comes to tracking expenses and budgeting. Despite advancements in artificial intelligence (AI) and automation, many budgeting apps still require **manual data entry** or calculations, resulting in an inefficient and potentially inaccurate process. This inconvenience often leads users to poorly manage their budgeting or abandon it altogether, hindering their ability to optimize spending habits and achieve their financial goals.

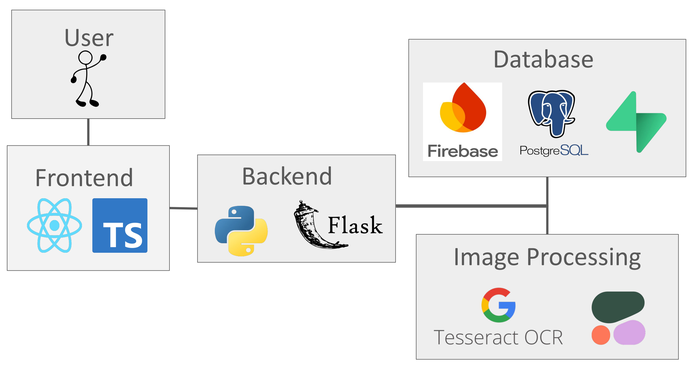
The purpose of Plutos is to **streamline expense tracking** by enabling users to log their expenses simply by taking photos of receipts. Users can then **visualize their spending habits** over time, gaining insights into how their expenses align with their income and budget progress. By automating much of the tracking process, Plutos helps users manage their finances more efficiently and stay on track with their goals.

Figure 1. Survey results for how participants track their expenses.

*"I currently use a budget template excel spreadsheet which has been pretty good so far but I still have to **manually input** my entries every time, which can be very **time consuming**"*

- Participant from interview

Engineering Design



Key Features

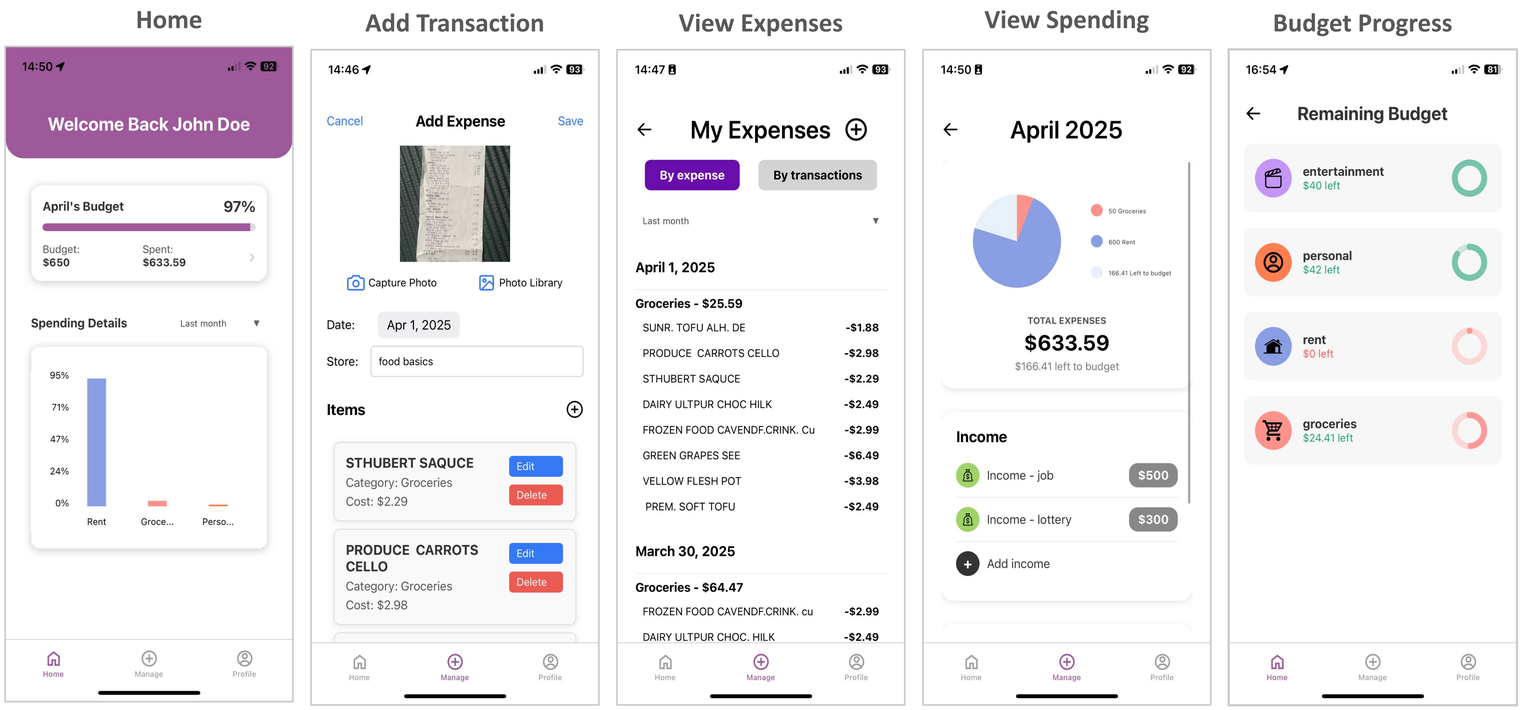
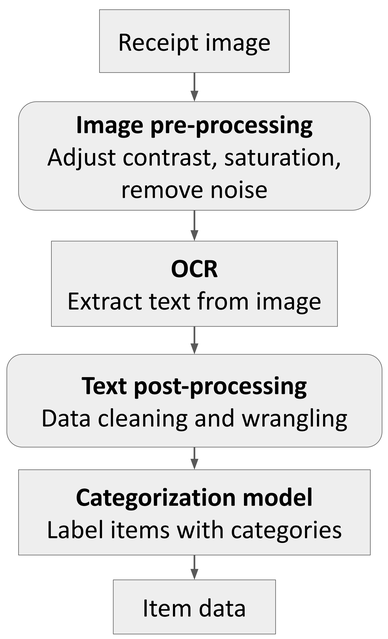


Image Processing



Key Benefits

- **Automated data entry:** The app reduces manual effort by automatically extracting item names, costs, and categories from receipt photos, ensuring a quicker logging process.
  - Changes can be mad, with autocomplete to suggest previously purchased items.
  - The image takes < 3 seconds to process, and the OCR and categorization models have 80% and 82.46% accuracy respectively, as reported in V&V Report.
- **Customizable budgeting goals:** Users can set personal budgets for each spending category, allowing for a financial plan suited to their needs and goals.
- **Fine-tuned category tracking**: Each item can be labelled with a category, enabling more fine-grained and accurate spending allocations.
- **Visualize expenses, incomes, and budgets:** Users can view how their spending relates with their income and budgets for each category over time.
- **Ease of use:** Simple, user-friendly interface with a low learning curve, making it accessible for users of all tech and financial levels.

1,008.94 KiB

Edit | Lock | Replace | Delete

CAPSTONE INTELLIGENCE

Capstone Project
Department of Computing and Software
Mechatronics & Software Engineering

on

eneral contractor is an antiquated ffect ve methods such as pencil and conversations. This leads to wasted which all u timately lead to more .

Android and iOS that will help the e i nsite(s) and better understand g term time frames.

cheduling and organization of onsite equ pment tracking.

verview

Tools Used

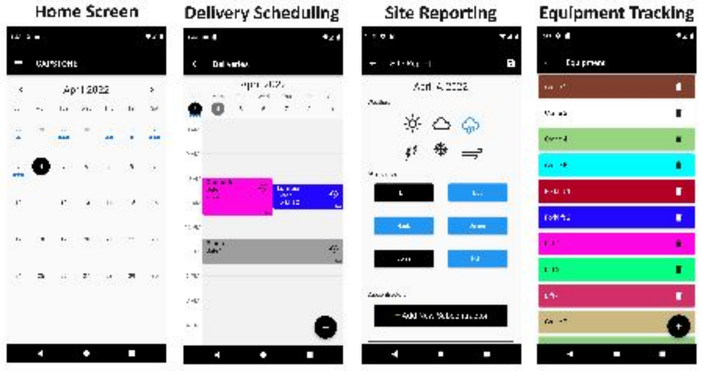
Firebase
Dart
Flutter
GitHub

, such as dialogs
ta entry, and app
out the app.

la Field Entry

Key Features

Home Screen | Delivery Scheduling | Site Reporting | Equipment Tracking



Key Benefits

- **Streamlined Communication:** Especially important for delivery scheduling and ensuring employees, supervisors, and equipment operators are aligned
- **Simplicity:** Focus on the desired use cases for construction site management
- **Central Cloud Information Storage:** One safe place where data & reports exist
- **Potential for Data Analysis:** Information collected via app can be used to perform analysis to derive insights, understand performance, track efficiency
- **Ease of Use:** User friendly, low learning curve, intuitive user experience
- **Transparency:** Increased visibility of onsite activities for workers and corporate

Acknowledgements

We would like to acknowledge and thank **Abdulla Jasim** (previously assistant site supervisor at El is Don) for his instrumental role in helping our group define requirements and imagine the user experience.

Our Team

CAPTURING MATHEMATICAL KNOWLEDGE IN DRASIL: THE CASE OF THEORIES

Jason Balaci (balacij), Dr. Jacques Carette (carette)
Department of Computing and Software, McMaster University

McMa
Universit

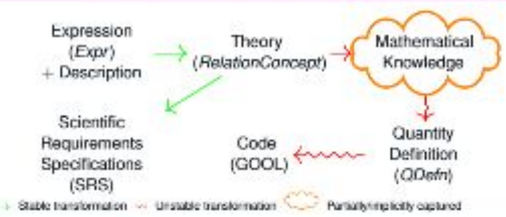
What is Drasil?

Drasil is a framework for generating families of software artifacts from a coherent knowledge base, following its mantra, "Generate All The Things!". Drasil uses a series of variably sized Domain-Specific Languages (DSLs) to describe various fragments of knowledge that domain experts and users alike may use to piece together fragments of knowledge into a coherent "story". Through forming some coherent "story" in a domain captured by Drasil, a representational software artifact may be generated. Drasil currently focuses on Scientific Computing Software (SCS), following Smith and Lai's Software Requirements Specifications (SRS) template as described in [4]. Behind the scenes of the SRS, a mathematical language is used to describe various theories, and have representational software constructed via compiling to Generic Object-Oriented Language (GOOL) [2]. Through encoding knowledge in Drasil, an increase in productivity (and maintainability) in building reliable and traceable software artifacts is observed [5], specifically in SCS [3]. Drasil's source code (Haskell), case studies, and documentation studies can be found on its website; `https://jacquescarette.github.io/Drasil/`.

Research Motivation/Problem

- *Theories* are constructed using a natural English description and a single term from a single universal mathematical expression language.
  - Expressions must be precisely written in a manner "digestible" to the code generator so that a representational software code snippet can be constructed.
  - Not all expressions have definite values and are immediately usable in all programming languages.
- Only a handful of the case studies generate code, because...
  - Understanding of the expressions is weak and brittle as they don't expose sufficient information about the theories.
  - Software artifacts are validated, we must obey rigid rules of other languages.
  - Cognitive load of writing expressions in precise manners to accommodate the code generator would increase.

Mathematical Knowledge Flow



Capturing Mathematical Knowledge

What changed?

- More static validation! Safer generation!
- Expression language division: *Expr* ⟶ *Expr'* ∪ *ModelExpr* ∪ *CodeExpr*
  - *Expr'*: Restricting the language to terms "well-understood", with a definite meaning and value.
  - *ModelExpr*: Restricting the language to terms with definite meanings, but not necessarily definite values.
  - *CodeExpr*: Restricting the language to terms that have a definite meaning and value to most general purpose programming languages, with some goodies for OOP.
- Created Typed Tagless Final (TTF) [1] encodings; Expression creation is just as easy as it ever was!

- Created a system of classifying *theori*
  - Increased the depth & breadth of kn
  - First-class representation of theorie components fully exposed. No mor sion of mathematical expressions (understood pieces of knowledge (hi
  - Instances of theories usable in a wi statically & reliably checked for valid
  - Creating instances of theories c editor-like ease.
- Improving productivity, stability, and fle
- Current theory types:
  - *EquationalModel*: $x = f(a, b, c, \ldots)$
  - *EquationalRealm*: $x = a \vee x = b \vee$
  - *EquationalConstraints*: $a \wedge b \wedge c \wedge$
  - *DEModel*: $\ldots \frac{dy}{dx} \ldots$

Conclusion & Futu

- Through capturing and classifying m we earn significant gains in flexibility, u
- *In progress*: Adding extra type inform lowing us to add type information to GO static validity rules of various mathem
- Teaching more theories to Drasil!

Reference

(1) ... Jacques Carette, Oleg Kiselyov, and Chung-chieh Shan. "... ...ciagol: interpreters for Gallina typed languages". In: Journal ... pp. 509–543.

(2) Jacques Carette, Brooks MacLachlan, and W. Spencer Smi ... group". In: Proceedings of the 25th ACM SIGPLAN Workshop ... Nov. 2016), pp. 49–60.

(3) W. Spencer Smith. "Beyond Software Carpentry". In: 2018 IE ... ing for Science (held in conjunction with ICSE'18). 2018, pp. 60–6

(4) W. Spencer Smith and Lei Lai. "A New Requirements Templ ... ings of the First International Workshop on Situational Requi ... Techniques and Tools to Support Situation-Specific Requirements ... junction with 13th IEEE International Requirements Engineering ... 157.

(5) Daniel Szymczak, W. Spencer Smith, and Jacques Carette. ... proach to Scientific Software Development". In: Proceedings of 2 ... national Conference on Software Engineering (ICSE). Austin, Tex

examples from lecture slides