Software Engineering 2

Project Title:

Personal Finance Tracker

Team Members:

Ragini Parag Kulkarni - 1002141368

Rajat Ramesh Dungarwal - 1002161059

Shubham Sunil Hingu - 1002161058

Nisarg Thakore – 100219943

1: Abstract

The Personal Finance Tracker is an Android application developed to help users effectively manage their financial activities. This project delivers a comprehensive solution for personal finance management, including income and expense tracking, budget management, savings goals monitoring, bill payment reminders, and financial reporting capabilities. Using an Agile development approach, our team successfully implemented core functionalities while maintaining high standards of security and user experience. The application integrates modern technologies including Android SDK for frontend development, Flask for backend services, and MySQL for database management.

2: Project Initiation

Project Proposal

The Personal Finance Tracker is an android application designed to help users manage their financial activities effectively. The project aims to provide comprehensive financial management capabilities including:

- Income and expense tracking
- Budget management
- Bill payment reminders
- Financial reporting and CSV export functionality

SDLC Model:

We are using Agile SDLC model. It gives emphasis on continuous testing, iterative development, and regular feedback through practices like sprints and user acceptance testing (UAT). Agile allows for flexibility in responding to changes and promotes collaboration within the team, making it ideal for our project Personal Finance Tracker.

Team Members and Roles:

- Ragini Parag Kulkarni (1002141368): UI Design Lead & Frontend Development
- Rajat Ramesh Dungarwal (1002161059): Budgeting Tools & Frontend Integration
- Shubham Sunil Hingu (1002161058): Backend Services & Database Design
- Nisarg Thakore (100219943): Testing & Deployment Lead

Feasibility Study

Technical Feasibility:

- Frontend: Android SDK provides a native development environment for Android apps, ensuring high performance and usability
- Backend: Python's Flask is a lightweight framework that allows for quick and flexible development of web applications.
- Database: MySQL offers reliable and structured data storage with robust querying capabilities.
- All chosen technologies have strong community support and documentation
- Team members possess required technical skills

Operational Feasibility:

- Market demand exists for personal finance management tools
- User-friendly interface ensures easy adoption
- Mobile accessibility meets user preferences
- Automated notifications enhance user engagement

Economic Feasibility:

- Development uses open-source technologies reducing licensing costs
- Cloud hosting provides scalable infrastructure
- Development timeline of 7 weeks is achievable with current team size
- Estimated budget of $471,400 covers development and deployment costs

Cost Estimation

Using the COCOMO model:

1. **Development Time Constraint**: Project deadline is November 10, 2024, allowing 1.75 months for development and testing with 10000 LOC.
2. **Effort Distribution**:
    - Total effort calculated using COCOMO model: 25.68 person-months.
    - With 4 team members, normal completion time is approximately 6.42 months.
3. **Acceleration Factor**:

    - To meet the 1.75-month deadline, the team needs to work 3.67 times faster than normal.
4. **Revised Effort per Person**:
    - Normal effort per person: 6.42 person-months.
    - Revised effort per person due to acceleration: approximately 23.57 person-months.
5. **Revised Total Cost**:
    - Assuming an average salary of $5000 per person-month, the total cost is $471,400.

Detailed Budget Breakdown:

- **Team Size**: 4 people
- **LOC**: 10000
- **Original Effort**: 25.68 person-months
- **Time Available**: 1.75 months
- **Effort per Person**: 23.57 person-months
- **Average Salary**: $5000 per person-month
- **Total Cost**: $471,400

Quality Assurance

- Defined Quality Standards: Establish quality criteria for deliverables from the outset.
- Continuous Testing: Implement automated testing throughout development using frameworks like Jest.
- Code Reviews: Regularly review code to maintain standards and best practices.
- User Acceptance Testing (UAT): Engage end-users for feedback to validate functionality at critical milestones.

Quality Factors

1. Functionality:

- Suitability: The app should offer appropriate functionality for personal finance management, including expense tracking and budgeting.
- Accuracy: The accuracy of all financial calculations and transactions is essential for accurate financial reporting.
- Security: User data should be protected by secure authentication and authorization.

2. Reliability:

- Maturity: Performance and bugs should be absent from the app.
- Recoverability: Ensure that backups and restores of financial data can be achieved.

3. Usability:

- Operability: It should be user-friendly with smooth navigation, so that users with varying levels of tech knowledge can find it easy to use.

4. Maintainability:

- Analyzability: An app should have well-organized and documented code for easy debugging, modification, or updating.

Risk Identification

1. Technical Risks
    - Android version compatibility issues
    - Flask scalability challenges
    - MySQL performance bottlenecks
2. Project Management Risks
    - Timeline constraints
    - Team communication challenges
    - Scope creep
3. Operational Risks
    - Data security vulnerabilities
    - User adoption challenges
    - System scalability issues

Risk Assessment

Risk Matrix:

Risk	Probability	Impact (1-5)	Probability × Impact
API Integration Issues	0.8	5	4
Timeline Constraints	0.8	5	4
Data Security	0.6	4	2.4
Performance Issues	0.5	3	1.5
Team Communication	0.2	2	0.4

Risk Mitigation Plan

Critical Priority:

1. Android Compatibility Issues
    - Comprehensive device testing
    - Minimum SDK version optimization
    - Regular compatibility testing
2. Timeline Constraints
    - Agile methodology adoption
    - Regular progress tracking
    - Priority feature implementation first

High Priority:

- Data Security
    - SQL injection prevention

- Secure API endpoints
- Encrypted data transmission

3: Project Scope

Scope Definition

The project includes tracking income and expenses, setting budgets, goal tracking for savings, bill reminders, and reporting capabilities.

Features

1. **Login/Signup functionality** for user authentication.
2. **Expense tracking** with categorization.
3. **Budget management** and notifications for upcoming payments.
4. **Reports** with CSV export options.

Constraints & Limitations:

Limited to Android devices, using MySQL for structured data storage.

Does not support multiple device types (Android only)

Target Audience:

Primarily individuals seeking organized financial tracking, with a user-friendly interface for various tech skill levels.

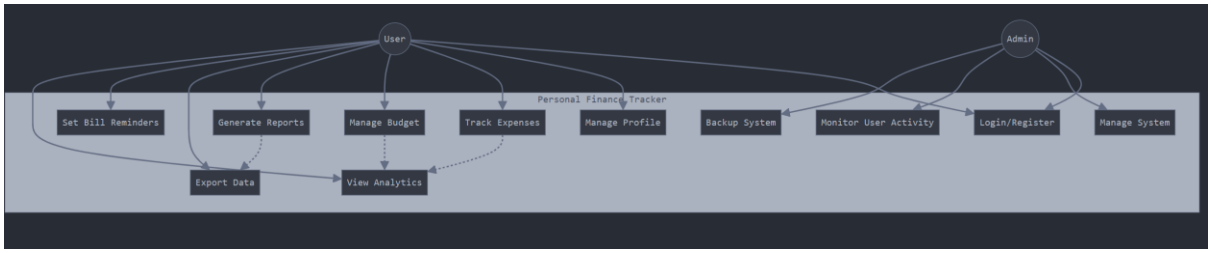
4: Requirements Analysis

Functional Requirements

1. User Authentication
    a. User registration and login
    b. Secure password management
2. Transaction Management
    a. Add/edit/delete transactions
    b. Categorize expenses
    c. Track income
3. Budget Management
    a. Create monthly budgets
    b. Track budget progress
4. Reporting
    a. Generate financial reports
    b. Export data to CSV

Non-Functional Requirements

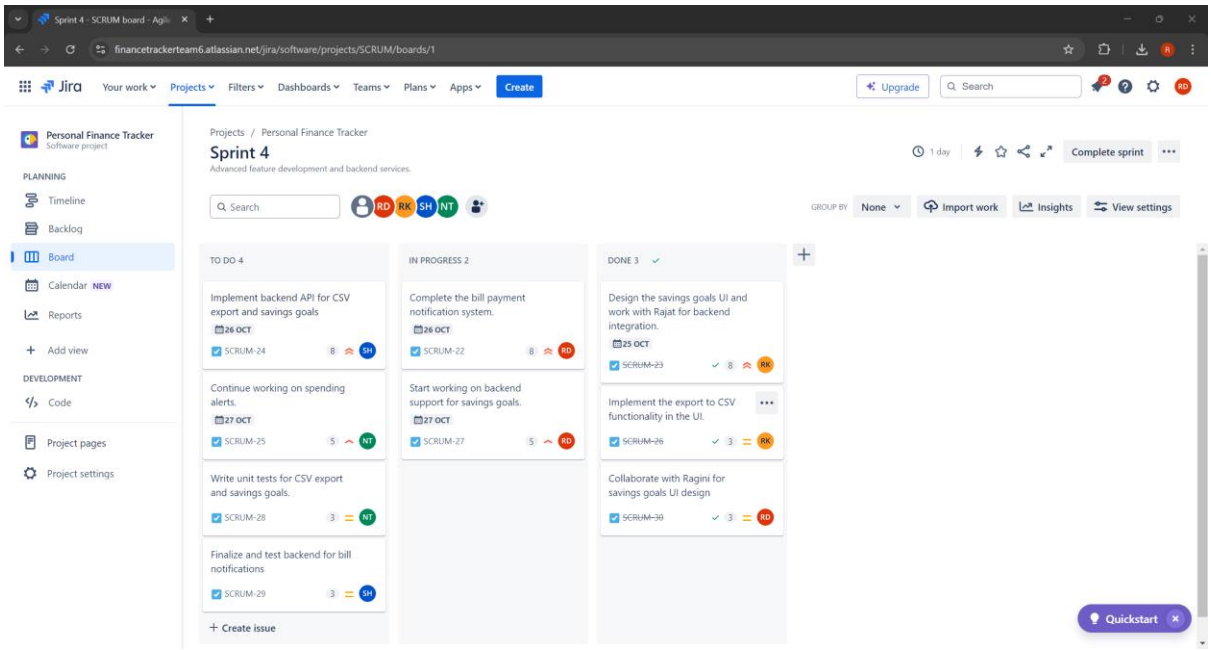
1. Performance
    a. App load time under 3 seconds
    b. Database query response under 3 second
2. Security
    a. Encrypted data transmission
    b. Secure API endpoints
3. Usability
    a. Intuitive user interface
    b. Responsive design

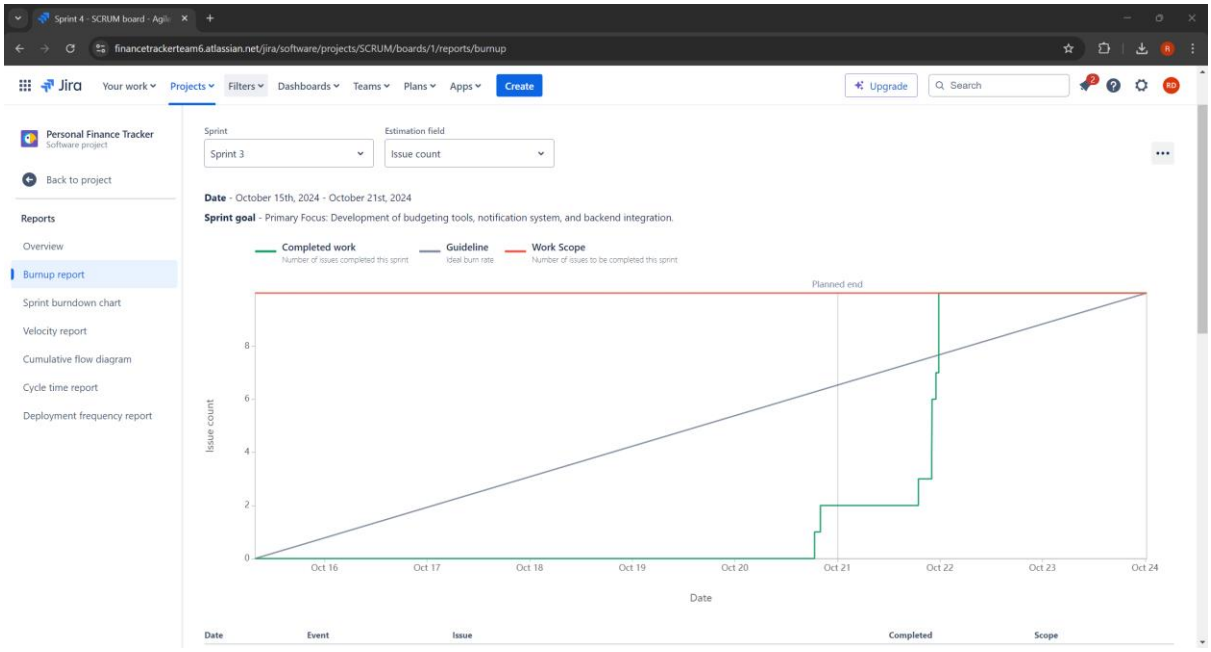


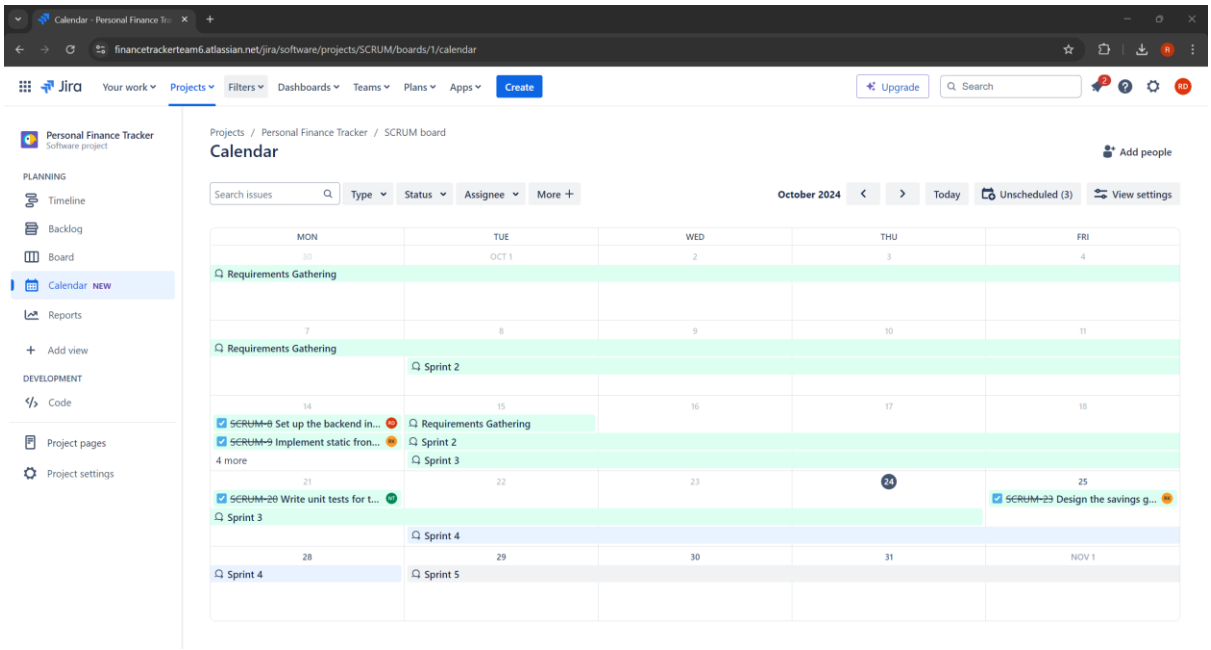
5: Project Management

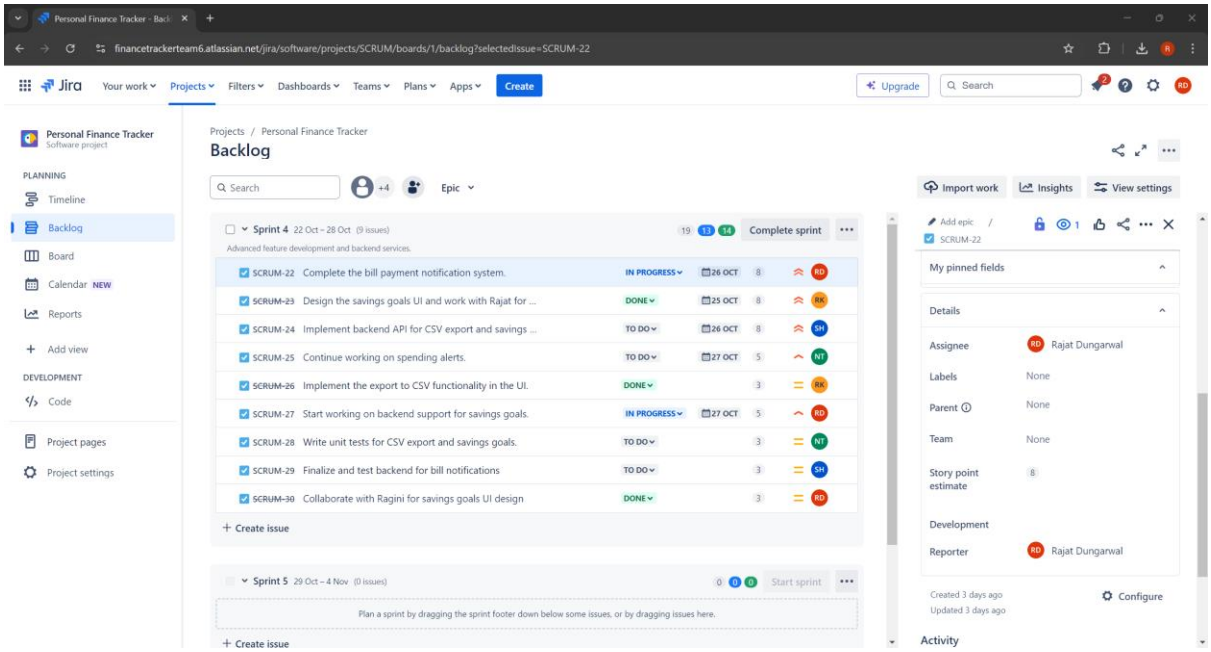
JIRA Implementation:

- Sprint planning and tracking
- Task assignment and monitoring
- Bug tracking
- Progress reporting



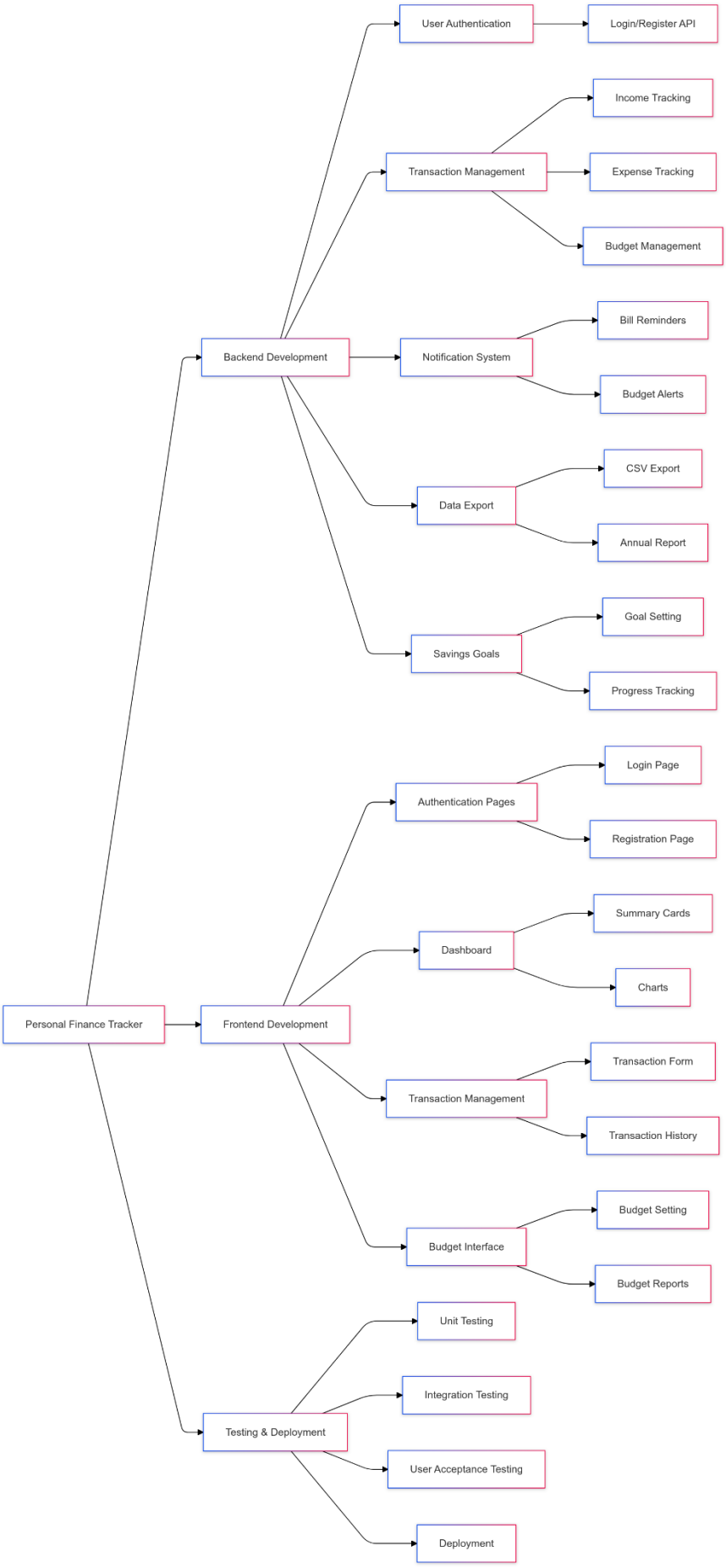

Sprint 4 - SCRUM board - Agile
financetrackerteam6.atlassian.net/jira/software/projects/SCRUM/boards/1/reports/burnup
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
Upgrade
Search
Personal Finance Tracker
Software project
Back to project
Reports
Overview
Burnup report
Sprint burndown chart
Velocity report
Cumulative flow diagram
Cycle time report
Deployment frequency report
Sprint
Sprint 3
Estimation field
Issue count
Date - October 15th, 2024 - October 21st, 2024
Sprint goal - Primary Focus: Development of budgeting tools, notification system, and backend integration.
Completed work
Number of issues completed this sprint
Guideline
Ideal burn rate
Work Scope
Number of issues to be completed this sprint
Planned end
Issue count
0
2
4
6
8
Oct 16
Oct 17
Oct 18
Oct 19
Oct 20
Oct 21
Oct 22
Oct 23
Oct 24
Date
Date
Event
Issue
Completed
Scope


Calendar - Personal Finance Tra
financetrackerteam6.atlassian.net/jira/software/projects/SCRUM/boards/1/calendar
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
Upgrade
Search
Personal Finance Tracker
Software project
PLANNING
Timeline
Backlog
Board
Calendar NEW
Reports
Add view
DEVELOPMENT
Code
Project pages
Project settings
Projects / Personal Finance Tracker / SCRUM board
Calendar
Add people
Search issues
Type
Status
Assignee
More
October 2024
Today
Unscheduled (3)
View settings
MON
TUE
WED
THU
FRI
Requirements Gathering
Sprint 2
SCRUM-8 Set up the backend in...
SCRUM-9 Implement static fron...
4 more
Sprint 3
SCRUM-20 Write unit tests for t...
SCRUM-23 Design the savings g...
Sprint 4
Sprint 5



6: Project Planning

Work Breakdown Structure


Personal Finance Tracker
Backend Development
User Authentication
Login/Register API
Transaction Management
Income Tracking
Expense Tracking
Budget Management
Notification System
Bill Reminders
Budget Alerts
Data Export
CSV Export
Annual Report
Savings Goals
Goal Setting
Progress Tracking
Frontend Development
Authentication Pages
Login Page
Registration Page
Dashboard
Summary Cards
Charts
Transaction Management
Transaction Form
Transaction History
Budget Interface
Budget Setting
Budget Reports
Testing & Deployment
Unit Testing
Integration Testing
User Acceptance Testing
Deployment

Project Schedule

Week 1-2 (Sept 24 - Oct 7)

- Requirements gathering
- Environment setup
- Database design
- Basic UI/UX design

Week 3-4 (Oct 8 - Oct 21)

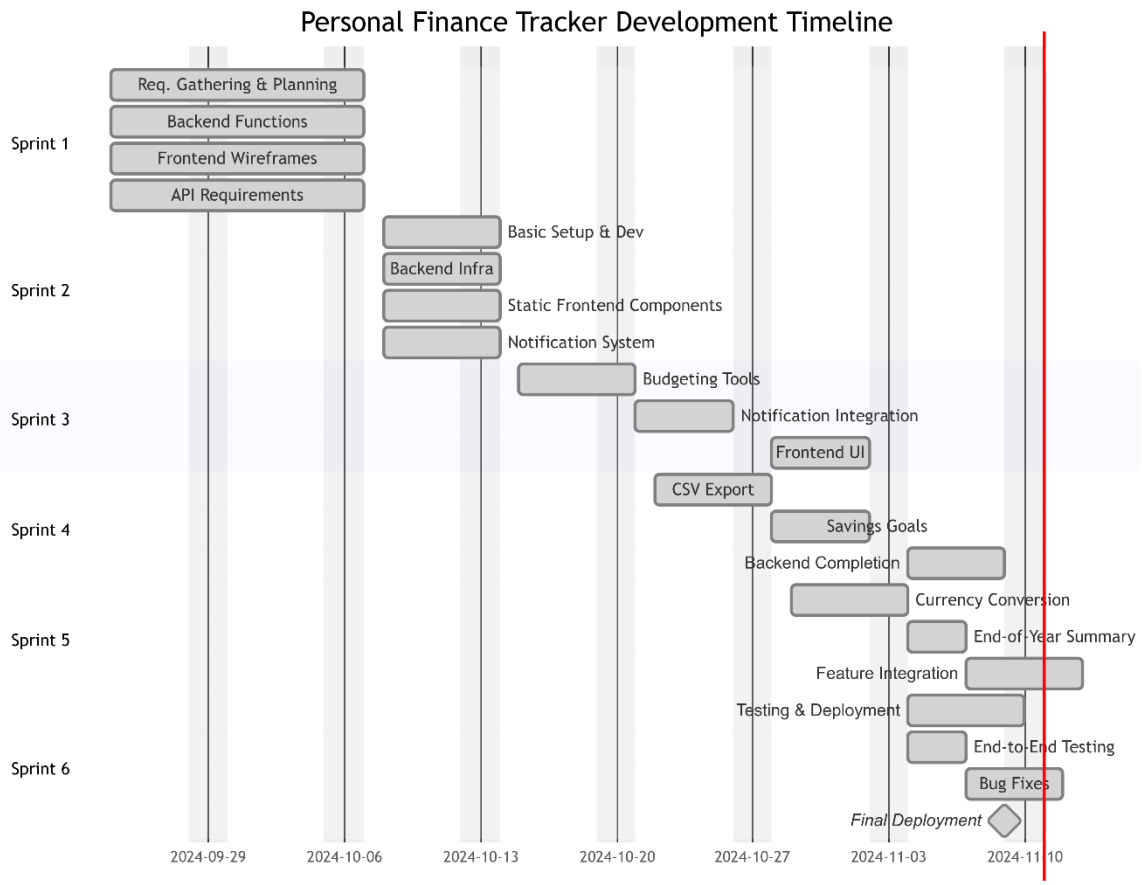
- Core feature development
- Frontend implementation
- Basic backend services
- Initial testing

Week 5-6 (Oct 22 - Nov 4)

- Advanced feature development
- Integration testing
- Bug fixes
- User acceptance testing

Week 7 (Nov 5 - Nov 10)

- Final testing
- Documentation
- Deployment
- Project submission



Grey blocks – Done (100% complete)

Resource Allocation

Human Resources:

1. Frontend Team (Ragini & Rajat)
    - Visual Studio Code
    - Android SDK
    - UI/UX Design Tools
2. Backend Team (Shubham)
    - Python's Flask
    - Database Management Tools (MySQL)
    - API Testing Tools
3. Testing & Deployment (Nisarg)
    - Testing Frameworks
    - Deployment Tools
    - Monitoring Systems

Software Resources:

- Development: Visual Studio Code, Git

- Project Management: JIRA
- Database: MySQL
- Version Control: GitHub
- Testing: Jest, Testing Frameworks compatible with Flask (for backend)

Team Member Tasks:

Ragini Kulkarni:

- Lead UI design and development for income/expense tracking and transaction history.
- Design the financial reports UI and ensure it integrates well with backend data.
- Collaborate with Rajat on the overall frontend development to ensure a seamless user experience.

Rajat Ramesh Dungarwal:

- Lead the development of the budgeting tool, including UI and backend integration.
- Implement the bill payment notification system (for recurring bills like rent, utilities, etc.).
- Collaborate with Ragini on the frontend development to ensure seamless user experience across all features.
- Collaborate with Ragini on API integration between frontend and backend (for income/expense, and notifications).

Shubham Hingu:

- Develop backend services for income/expense tracking, transaction history, and bill payment notifications.
- Implement CSV export functionality and manage core API development.
- Oversee database structure design for optimized performance.

Nisarg Thakore:

- Focus on developing spending alerts and optional currency conversion.
- Conduct unit testing for feature stability.
- Manage deployment with a focus on ensuring app scalability.

7: Design and Architecture

Presentation Layer (Frontend)

- Android native application
- XML-based layouts for responsive UI
- Activity/Fragment-based navigation
- Data binding for real-time updates

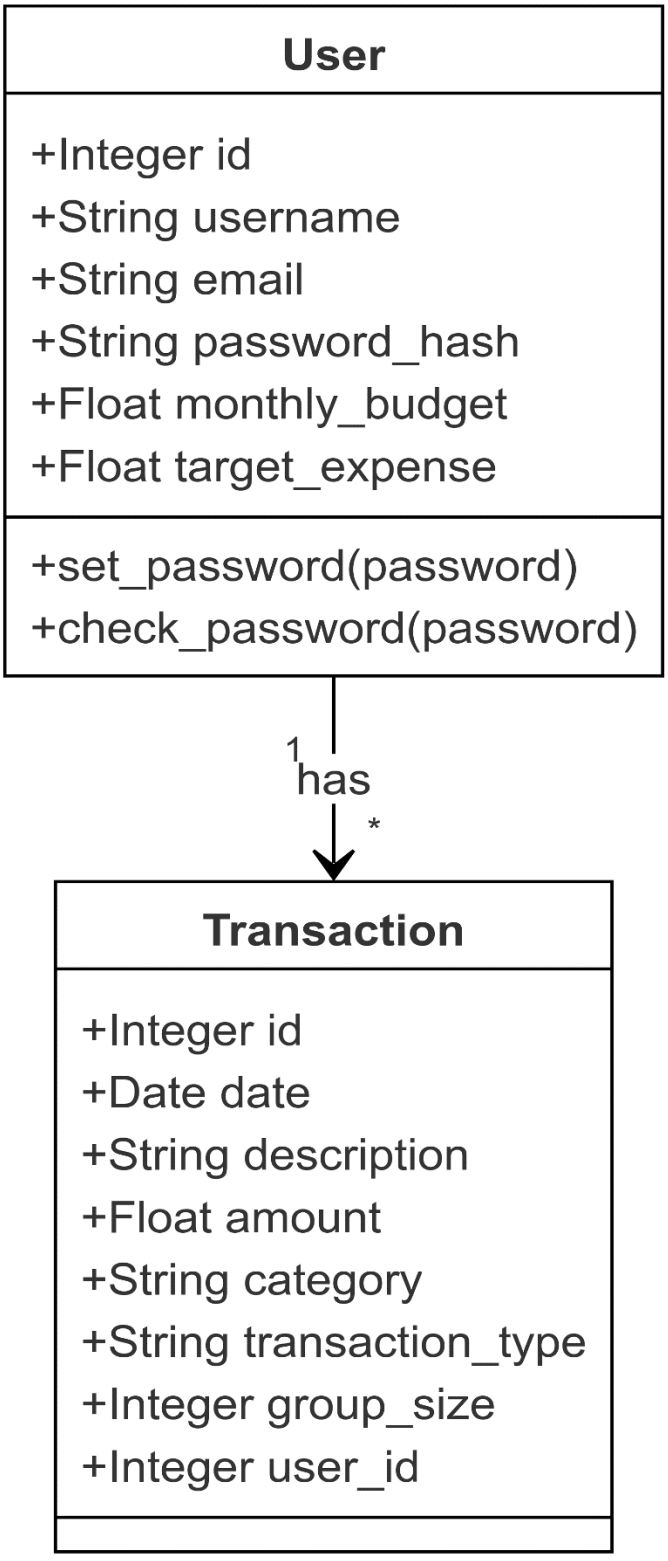
Application Layer (Backend)

- RESTful API services using Flask
- Business logic implementation
- Data validation and processing

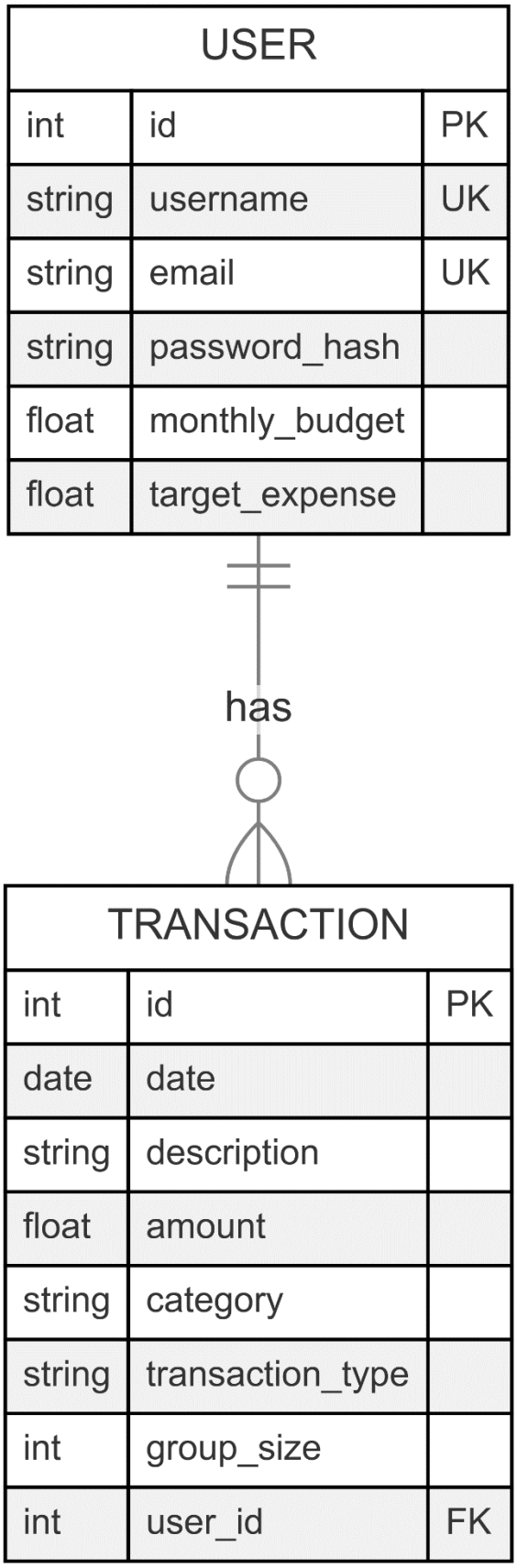
Data Layer

- MySQL database for persistent storage
- SQLAlchemy ORM for database operations
- Cached data management
- Data backup and recovery systems

UML Diagrams


User
+Integer id
+String username
+String email
+String password_hash
+Float monthly_budget
+Float target_expense
+set_password(password)
+check_password(password)
1
has
*
Transaction
+Integer id
+Date date
+String description
+Float amount
+String category
+String transaction_type
+Integer group_size
+Integer user_id

ER Diagram


USER
int
id
PK
string
username
UK
string
email
UK
string
password_hash
float
monthly_budget
float
target_expense
has
TRANSACTION
int
id
PK
date
date
string
description
float
amount
string
category
string
transaction_type
int
group_size
int
user_id
FK

8: Implementation

Frontend Development

- Android Studio 2023.1.1
- Java Development Kit (JDK) 17
- Android SDK (API Level 21+)
- Gradle Build System

Backend Development

- Visual Studio Code
- Python 3.9+
- Flask 2.0.1
- MySQL
- Postman for API testing

Version Control

- Git/Github.

Third Party Libraries

- blinker==1.8.2
- Flask==2.0.1
- Flask-Cors==5.0.0
- Flask-Login==0.5.0
- Flask-Mail==0.9.1
- Flask-SQLAlchemy==2.5.1
- PyMySQL==1.0.2
- SQLAlchemy==1.4.31

Major Modules

- **Transaction Management**:
  - add_transaction: Adds a new transaction based on JSON input data.
  - get_transactions: Retrieves transactions for a specific user.
  - update_transaction: Updates an existing transaction if the user is authorized.
  - delete_transaction: Deletes a specified transaction if the user is authorized.
- **User Profile Management**:
  - get_user_profile: Retrieves the profile details of a user.
  - update_user_profile: Updates user profile details such as budget and target expense.

- **User Signup and Login**:
  - signup: Registers a new user, ensuring a unique username and email.
  - login: Authenticates a user with username and password, starting a login session.
- **Session Management**:
  - logout: Logs out the current user (HTTP method updated to POST).

- check_auth: Checks if a user is authenticated and retrieves their details if so.

app > models.py > Transaction

```python
from . import db
from flask_login import UserMixin
from werkzeug.security import generate_password_hash, check_password_hash

class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(128))
    monthly_budget = db.Column(db.Float, default=0.0)
    target_expense = db.Column(db.Float, default=0.0)
    transactions = db.relationship('Transaction', backref='user', lazy='dynamic')

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)

class Transaction(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    date = db.Column(db.Date, nullable=False)
    description = db.Column(db.String(200))
    amount = db.Column(db.Float, nullable=False)
    category = db.Column(db.String(50))
    transaction_type = db.Column(db.String(10))  # 'Personal' or 'Group'
    group_size = db.Column(db.Integer)  # Number of people in the group, null for personal transactions
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
```

Implementation Challenges

1. Frontend-Backend Integration
    a. Inconsistent data formats between Android frontend and Flask backend
    b. Handling asynchronous API calls and response management
    c. Error handling across different layers of the application
    d. Managing state synchronization between frontend and backend
2. Database Integration
    a. Managing database connections efficiently
    b. Handling concurrent transactions and race conditions
    c. Ensuring data consistency across multiple operations
3. Session Management
    a. Maintaining user sessions across application
    b. Managing session cleanup and timeout.

9: Testing

Testing Strategy:

1) Unit Testing:
    - Frontend Component Testing
    - Backend Service Testing
    - Database Operation Testing
2) System Testing

- Performance Testing
- Security Testing
- Compatibility Testing

Unit Testing

Test Case ID	Description	Expected Result	Actual Result	Status
1	Validate user login with correct credentials	User successfully logs in and redirects to dashboard	User successfully logged in	PASS
2	Validate user login with incorrect password	Show error message "Invalid credentials"	Error message displayed	PASS
3	Register new user with valid information	User account created successfully	Account created	PASS
4	Register user with existing email	Show error "Email already exists"	Error displayed	PASS
5	Add new expense transaction	Transaction saved to database	Transaction saved	PASS
6	Add transaction with negative amount	Show error "Amount cannot be negative"	Error displayed	PASS
7	Create monthly budget	Budget saved successfully	Budget saved	PASS
8	Set budget with zero amount	Show error "Budget amount must be greater than zero"	Validation failed	FAIL

System Testing

Test Case ID	Description	Expected Result	Actual Result	Status
1	App load time measurement	Load within 3 seconds	Loads in 2.8 seconds	PASS
2	Database query performance	Queries complete within 1 second	Average 0.8 seconds	PASS
3	Test on Android 10 device	App functions correctly	All features working	PASS

4	Test on Android 11 device	App functions correctly	All features working	PASS
5	Test on Android 12 device	App functions correctly	UI scaling issues	FAIL
6	Test SQL injection prevention	All injection attempts blocked	Successfully blocked	PASS

Solutions Implemented:

1. Standardized error handling across all layers
2. Token-based authentication with refresh mechanism
3. Connection pooling for database optimization
4. Proper session state management

10: Maintenance

1. Regular Updates
    a. Monthly security patches
    b. Quarterly feature updates
    c. Bug fix releases as needed
2. Monitoring
    a. Performance monitoring
    b. Error tracking
    c. User feedback collection
3. Scalability Considerations
    a. Database optimization
    b. API performance monitoring

11: Conclusion

The Personal Finance Tracker project successfully delivered a comprehensive mobile application for personal finance management. The team effectively utilized Agile methodology to manage development challenges and meet project deadlines. While some features like currency conversion and advanced reporting were completed in later sprints, the core functionality was delivered with high quality and security standards.

12: References

1. Android Developer Documentation
2. Flask Documentation (2.0+)
3. MySQL 8.0 Reference Manual
4. Material Design Guidelines
5. Jest Testing Framework Documentation
6. PyTest Documentation