SW Mini Project

EC463

Objective

- Agile SW development
- Code management (GitHub)
- Learn how to use REST APIs
- Design your own REST API
- Utilize cross-platform environment to build an application

Alert…Alert…Alert

- If you like application development, this is addictive! I believe it is a cool app
- You should start now and not wait till last minute
- Spend few hours a day
- ***I Did Not Implement Any Of It.*** I just investigated the APIs.
- Lots of debugging

SERIOUSLY! It is fun!

Mini Project User Stories

Mood Chat Application

User:

- A social media savvy person

Required User Stories

- I login to my application using my Google Email account
- I can discover everyone in the system by searching for their name or email address
- I can write messages to anyone I chose

Nice to have User Stories

- Private Chat rooms where an admin can invite people to the room.
- Block certain users for writing you messages
- Sentiment analysis (positive, negative or neutral) of all messages
    - Order and/or filter per sentiment analysis

Application

- Could be WEB app, Mobile app, Python app
- Backend could be of your choice

Homework Requirements

- Two-person teams
- Follow Agile SW development
- SW and documentation are in GitHub
  - Create project GitHub
  - If you have not used GitHub, please go through the Tutorials
  - Each person must contribute to the project. You cannot work on one computer and have “team” merges. Each person must contribute
- Document your design decisions in GitHub
- Document your REST APIs in GitHub (use GitHub Wiki)
- Testing and results must be documented in GitHub
  - Bugs and tasks to be documented in GitHub Issues
- Upload screenshots and Videos of your final project on GitHub
- Extra credit if you learn and use GitHub Actions

LLM and CoPilots

- Yes for this exercise, you can use ChatGPT or any Copilot
- Condition: Acknowledge that and show what your CoPilot helped you with
- No deduction points for this, on the contrary, it may allow you to do way much more!
- Recall you will debug and test. Some APIs and recommendations may have changed since 2021 :-)

Helpful Links

1. What is Agile?

Agile Development

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly., Atlassian.com

Manifesto

We are uncovering better ways of developing

software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Principles behind the Agile Manifesto

Scrum and Kanban approaches

GitHub and Code management

Please, please please pay attention to

- Everyone in the team should be part of the GitHub project
- Everyone should contribute to the project
- PLEASE PLEASE PLEASE: Dont work from one computer and account!

Please don’t store keys or any confidential information in GitHub

Use GitHub Secrets
For Google Cloud, you can follow tutorials
Or make it as part of your deployment parameters

Cloud Services

Use Free Tier Cloud Services

- AWS
- Azure Free For Students
- Google Firebase Free Limits
- Heroku Free Limits

Cross Platform Application Frameworks

They are also called

Multi-Platform Application Frameworks

Cross-Platform Application Software

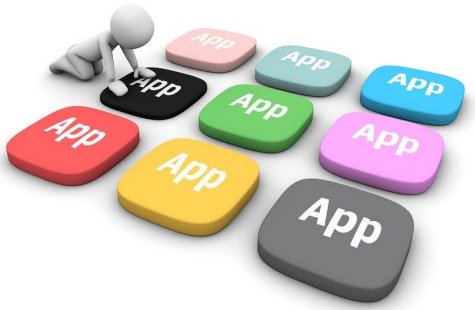
Multi-Platform Application Software

Platform-Independent Application software

Client Frameworks

A framework is a library that offers opinions about how software gets built. These opinions allow for predictability and homogeneity in an application; predictability allows software to scale to an enormous size and still be maintainable; predictability and maintainability are essential for the health and longevity of software

Cross Platform Frameworks



Framework Engine

I/F to iOS Native Modules and APIs

I/F to Android Native Modules and APIs

iOS

android

Pros and Cons of Cross Application Frameworks

	Browser	**Cross Platform**	**Native**
Performance	Worse	Better than browser and depends on framework used	Best
Ease of Development (e.g, reuse of code, coding skills)	Easiest	Except for OS or HW plugins, rest is reusable	Hardest and least code reuse
Consistency of Application	Consistent	Consistent	Harder to be consistent
Flexibility with regard to innovative features	Have to wait for new features	It is cumbersome but you can build plugins to frameworks. It can be sub-optimal.	Most flexible

Popular Mobile Cross Application Frameworks

- REACT Native by Facebook
- Flutter by Google
- Cordova by Adobe
- IONIC by Drifty
- Xamarin by Microsoft
- Titanium by Appcelerator
- Unity by Unity Technologies

Comparison

	REACT	**Flutter**	**Cordova**	**IONIC**	**Xamarin**	**Unity**
Company	Facebook	Google	Adobe	Drifty	Microsoft	Unity Technologies
Approach	Compile to Native	Compile to Native	Using WEBView	Using WEBView	Compile to Native	Compile to Native
Application programming Language	Javascript	DART	HTML Javascript	HTML Javascript	C# .net	C#
Community	Facebook, Instagram, Airbnb, Tesla, Skype: https://reactnative.dev/showcase	Google Alibaba	https://cordova.apache.org	https://showcase.ionicframework.com/apps/top	https://dotnet.microsoft.com/apps/xamarin/customers	https://unity3d.com/games-made-with-unity

References

- Server-side web-frameworks by Mozella
- Introduction to client-side frameworks by Mozella
- Web Frameworks Python
- node.js
- The Ultimate Guide to Learn Web Application Architecture

Twitter and application frontend and backend platforms

NLP and sentiment Analysis Resources

- I recommend Google NLP
- You can use OpenAI or any other

Suggestions

Suggestions

- You can user any environment you like: Azure, AWS, Google Cloud, etc. or host your environment .
- I recommend Google Firebase.
- I am recommending using Flutter or REACT native to build your application.
  - Firebase for Flutter iOS
  - Firebase for Flutter Android
  - Firebase for iOS Native
  - Firebase for Android Native
  - Firebase for REACT Native (Tutorial)
- REACT Firebase hocks

Hints: One person in each track

APP HINTS

- Hint 1: Setup your REACT Native Environment
    - Please note: If you have an iOS device, you need a Mac computer. If you don't have a Mac, please build it as an Android on the emulator.
- Hint 2: Go through REACT native Tutorial and build and run the Hello Applications
- Hint 3: Setup your Firebase (react-native-firebase). Setup authentication, database, and storage

Twitter HINTS

- Hint 1: Creating your developer account and apply for keys. This takes time.
- Hint 2: Check Twitter APIs for Google Cloud
- Hint 3: Get Familiar with Firebase

Milestones suggestions

- Phase 1:
  - REACT app with Google Login
- Phase 2:
  - Backend
    - Build a chat module with REST API
    - Test your chat APIs using Postman and/or Python
  - Application
    - Build the wireframes for chat application
- Phase 3:
  - Integrate backend with frontend and test

Then Go for the stretch goal! Or keep working on it :-)

Thank you