Team5 Ghost Squad

Final Presentation

Team5 USC Ghost Squad

Focus on Reusable Code
Multi-Generational
Software Development.

BY AREA OF CONTRIBUTION

UI – REACT(LILY, JACKSON, JOHN)
UI – IONIC(JOHN)
UI - HTML(PRAVANI JOHN)
BACKOFFICE (CALEB, CAROLINA, JOHN)
HOSTINGOPS(JOHN)

DEMOGRAPHICS

3 GRAD STUDENTS – 2 IN ABSENTEA
(3 UNDERGRADS IN ABSENTEA)
1 CAPGEMENI RESOURCE

Agenda

01	02	03	04	05	06
THE MATRIXED TEAM & WORK PRODUCTS	ARCHITECTURAL OVERVIEW BACK TO FRONT	FEATURES COMPARISON & CODE ANALYSIS	LIVE DEMO	LESSONS LEARNED	Q&A

About Me
Hosting/Repo Environment (Github/AzureDevops)
2 Min Demo (Ridefinder)
2 Min Demo (Parks Views - > Cart Differences)
2 Min Demo (Manager)
Slides and Questions. (Basics) -> Extended At Your Leisure.

About Me

John S. Stritzinger III
PMP & Global IT Architect

HISTORICAL

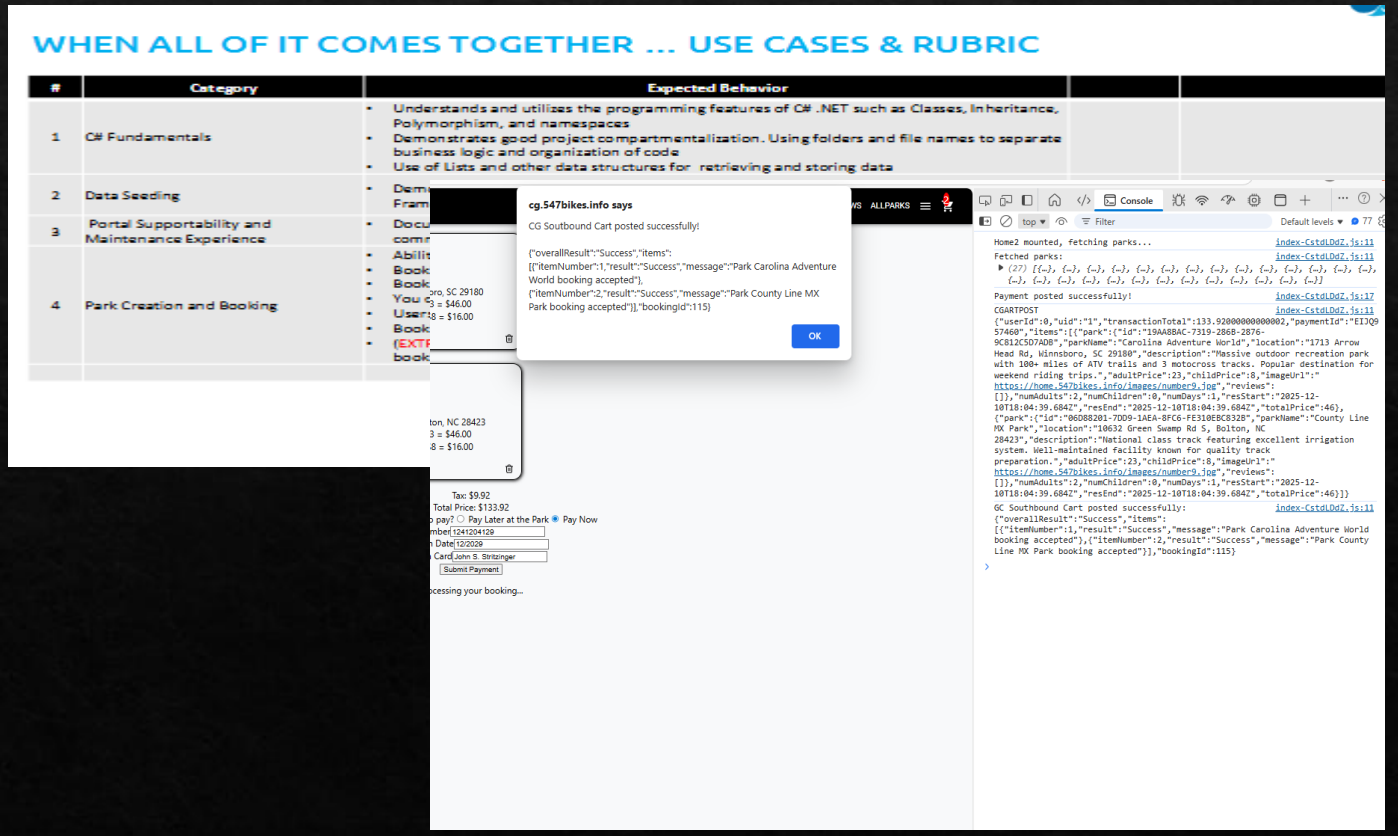
- Experienced Developer with 4 previous CIO Jobs for Public Companies & C-Level Public Officer for 20+ Years.
  - CIO DPNet/TimesMirror Newspapers
  - CIO Sales Operations for Cincinnati Bell
  - CIO Customer Operations for Cincinnati Bell
  - CIO Bank of America Network Services & CTO For the Corporation
- Began my career as a Consulting VP & DBA (Clearview Software/DP) managing the Bank of America Account.
  - Built the Business Card, Speciality Lending, and Home Equity Programs for then MBNA Now Bank of America Consumer, Business Card BU's.
- Family Business Acquired By Times Mirror & IXC Communications Subsidiary -> Managing Director of Product Development & BD Groups
  - Sold to Charter-> Cable & Video Specialist for Charter/Vyvx Video -> Managing Director Special Construction & Product Support.
  - Sold to CinciBell -> ILEC Transport and ROW Engineer, Marketing, Special Const. - > Top 5 Officer of the Corporation
  - Sold to Centurylink/L3/Lumen Global Services-> Managing Director Operations & Direct Report to COO. -> Moved to CTO Office Post Merger.
- Chief Architect Bank of America Corporation & COO of Global Operations.
- Principal Architect – Verizon Federal Consulting – National Security Programs, US Courts, Dept of Energy Outsource Contract, Whitehouse Comms(Obama).
- Completed More than 30 M&A Transactions most in Integrative Perspective, some just in due diligience.

RECENT

- Chairman and CEO of Greenville Associates Consulting
- Chairman and CEO of Capitol Technology Solutions
- Teaching Assistant in the College of Engineering Fall 2025.

Compliance To Rubric

- In General We wanted to be Compliant to the Rubric and meet all general requirements....
- Those issues ARE.....



Source Tree & Reusable References

01- GS ACHIEVEMENTS

- FALL2024 -> MOVIES PROJECT
  - (MOVIES, REVIEWS, CART, MANAGER)
- SPRING25->CERT AUTOMATION
  - (CERTS, AUTH, OCR, FILE, USERPROFILE, AZURE SERVICES
- FALL2025-> G/A PARKS SUPERSET, CG DIRTBIKES SUBSET
  - (ADVANCED CART, ADVANCED UI FEATURES, ERP FLOW THROUGH)

01 – REUSABLE CODE (OLD)

- We are seeking to build only reusable code while here on campus.
- In the Fall of 2024, our team built:
  - Sales-> Reusable React Menus with Material UI Templates.
  - OPS-> A CLI Installer Supporting 2 SQL Servers(MySQL, SQLServer) now w SQLITE
  - SECURITY->A Semi-structured DataModel with ASPX Domain Security Overlay.
- In the Spring of 2025, our team built:
  - Sales -> A Complex User-Userprofile Administrator
  - OPS-> OCR, ServiceBus, Azure Bit Bucket
  - SECURITY-> An Advanced Administrator Tool to Manage Security Logs.

01 - REUSABLE CODE(NEW)

- Team5 Achievements This Term:
  - SALES-> (Advanced Cart Features, Tax Table Management, UserProfile Accounts Native)
  - OPS-> (New DEVOPS Model from Development(Linux) to Production(Azure), Notification, NodeActions Controller)
  - SECURITY->(New Security Center)

01 – MATRIXED REFERENCES

ROOKIE

- T1FALL2024-> JACKSON, JOHN
  - (MATERIAL UI TEMPLATES, TOPMENU, RROUTER CONFIGS, SEMISTRUCTURED DATA)
  - IMPACTS SIGNIFICANT REDUCTION IN TIME IN STRUCTURAL RE-ENGINEERING OF CG UI, SEEDING

SEMI-PRO

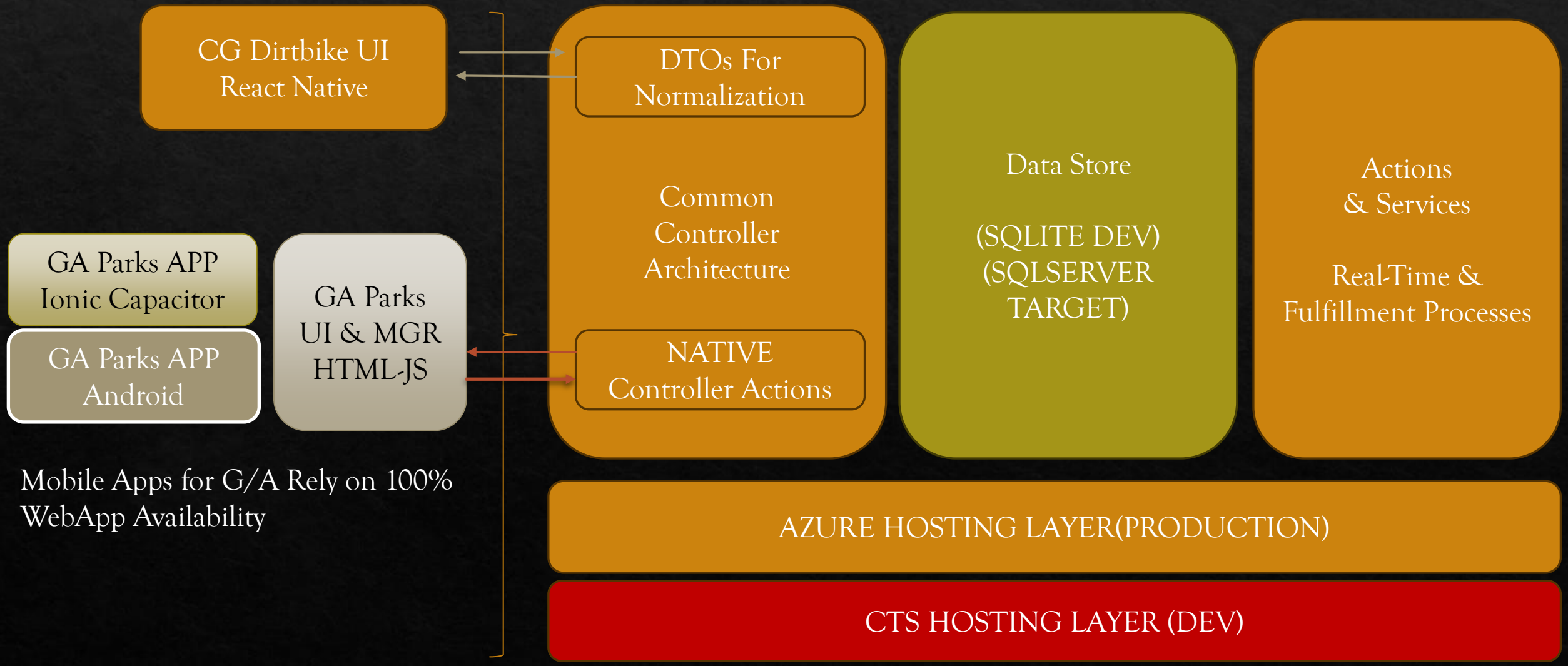
- T1SPRING2025-> CALEB, PRAVANI, CAROLINA, JOHN
  - (AUTH, CSS TEMPLATES FOR SITE, USERFLOWS, PROJMGMT, ADMIN PAGES, PLL, USER FEATURES, AZUREWEB)

PRO

- T1SPRING2025-> LILY(CG), JOHN
  - (REACT FRONT, IONIC FRONTPARKS, HOSTING, SECURITY, DTOS, SERVICES)

- MOVING FROM A ROOKIE- TO SEMI PRO- TO PROFESSIONAL IN 16 MONTHS
- NO DEGRADATION OF PERFORMANCE MOVING TO MATRIXED DELIVERY MODEL WITH HISTORICAL CONTRIBUTIONS.

02 Software Architecture



Graduate Features In Production.

1. Ionic React Project Using Capacitor and Android APK.
2. Driving Directions
3. Google Maps
4. Park Visualizations
5. User Auth Controller -> High Security
6. Azure Integration
7. 3 UIs vs 1 Required.
8. 33 Table X 4+ Endpoints vs < 20 Required for Minimal Operations.
9. Very Advanced Cart with UserProfile Extensions.
10. Very Advanced Pricing Controls with Custom Pricing Database Per Park.
11. Very Advanced Tax Model with Tax Tables for US and National.
12. Session Model
13. Notification via Sendmail Integration.
14. Previous 590 Class Features Ported to SQL Lite including File Controller, ServiceBus, and Azure Blob Storage.
15. Can Support Multi-Mode Auth types including Social Medial Logins.
16. Support G/A PLL Locking Functions.
17. Extremely Advanced Administrator Options For Users, Employees, Park Maintenance, Transactions, Refunds, Accounting.
18. Significantly more complicated Project Office, and Documentation Required for Dr. Valtorta and Dr. Valefar.

6Th Generation Star fighters

Team5 Ghost Squad

Delivery of Rubric Requirements Discussion

Shared Research a Requirement For MS? An Aside….

- I was told by the Executive Management Team in the CSCE department that I could not do a Thesis Project without each and every piece of our project being constructed on Premise.
- And …. Without Homegrown tools, and a Thesis… getting a Masters Degree is at least 2X harder and perhaps at least 1 possibly two years longer…?
  - The Issue is you have to take at least 15 7XX credits in some combination, and a thesis eats 9 of these… and can be stacked if you devote your life to something even over the summer…..
  - Otherwise you have to take at least 3 semesters or 1.5 years with 2 7XX classes which are extremely difficult
- Furthermore…without a Portal and Auth in place… any project you do is at a High School Level when you start and for at least six months thereafter….truly.

- I DECIDED TO TAKE THIS CLASS AS AN EXTENSION OF MY UNIVERSITY RESEARCH WHICH WAS APPROVED BY THE DEPARTMENT SO I COULD COME CLOSER TO FINISHING MY THESIS

The Ghost Portal…is now a 6th Generation Star Fighter…Spanning 18Mo, 4DBMS Platforms, 3 USC Teams & 18CH

Cap Gemeni Generations & JSS Upgrades

Gen1.0-Fall24->Cocky Movies
(MySQL)wReactNative & C# BackOffice

Gen1.5-Xmas24->MyLinkv1* (Sqlserver)
Gen1.9-Feb 25 ->MyLinkv2 Mobile & C# Temps (IonicReact)

Gen2.0-Spring25->CertAethelon
Gen2.5-Sum25->MyLinkv2** (Azure SQL) w Full Resume Support.

Gen6.0-Fall25->Ride Finder on SQL-Lite with Full Security Platform & ProReact

*CockyMovies Hosted & Migrated to SQL Server. Template EPs built
** CertAthelon Rebranded By USC with Campus Themes Resumes

USC Platform Migrations

- **Gen3.0 Spring25- ITIL Service Manager**
  - **Full ITSM and Network Manager(CSCE498)**
- **Gen4.0 Sum25 – Grid Hosting Manager**
  - **Hybrid Grid Deployment &Apps (CSCE498)**
- **Gen5.0 Sum(2)25 – EIS & HR Visualizations**
  - All on PostgreSQL Server v17
  - With V3,V4,V5 Ghost Portal and Auth
  - Option for Azure SSO Integration

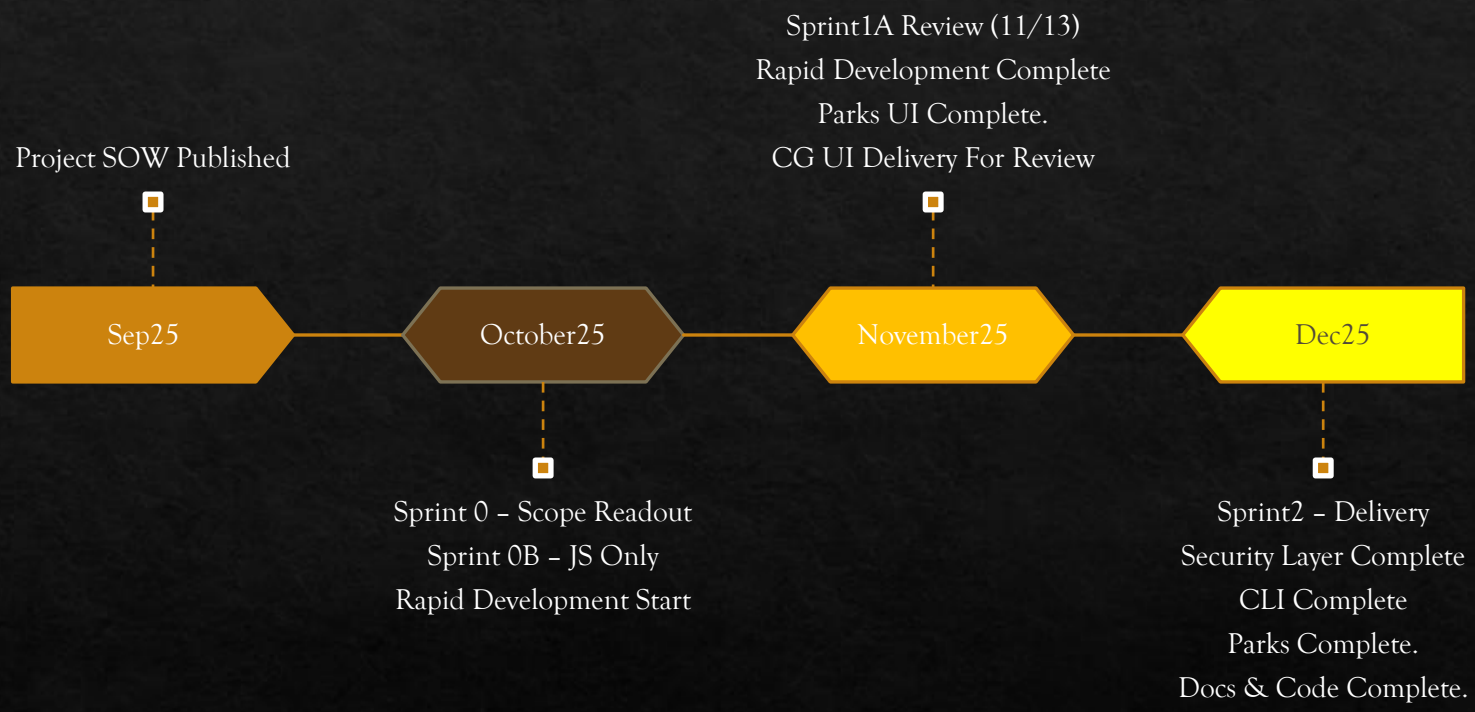
Capgemini

GhostPortal Futures (MS Research)

- Approved 798 Project #1 include -> AI Extension to ITSM/ITIL SERVICE MANAGER
- Approved 798 Project #2 include -> Adding End to End Application Layer Encryption BEFORE SSL.
- Approved Dissertation -> Grid Extensions for SSO over Hybrid Cloud.

- **WHY DOES THIS MATTER?**
  - We are integrating past work of various student teams along with updated research owned by the department.
  - Pravani, Sambit and I were department employees so the 590 Project, and my research this summer can be consolidated without risk to other factors according to Dr. Valtorta, and Dr. Valefar.
  - The Fall Team547 contributed only Material UI Skins which are opensource and had no other unique copyright or tradecraft. The Fall24 Backoffice was abandoned as not working, and the React Project on Axios didn't work in Production Hosting. The Fall24 Manager was however the precursor to the Spring25 Manager both of which I wrote.

Ride Finder Project – A Matrixed Complex Dev Effort



This Semesters Project is a Matrixed SOW due to smaller number of groups with less resources per group. The Majority of the Design of the UI was developed by Cap Gemeni Professional Developers, and the BackOffice we are designing in response to the SOW which ***is much more complex than previous semesters.***

Required Repos

RF.API-> GITHUB
RF.UI-> GITHUB
CODE ROLLUP-> GITHUB
STATICWEB->(RIDEFINDER)->AZUREDEVOPS
STATICWEB->(PARKS)->GITHUB

Extended Repos

RF.API-> GITHUB
RF.UI-> GITHUB
CODE ROLLUP-> GITHUB
STATICWEB->(RIDEFINDER)->AZUREDEVOPS
STATICWEB->(PARKS)->GITHUB

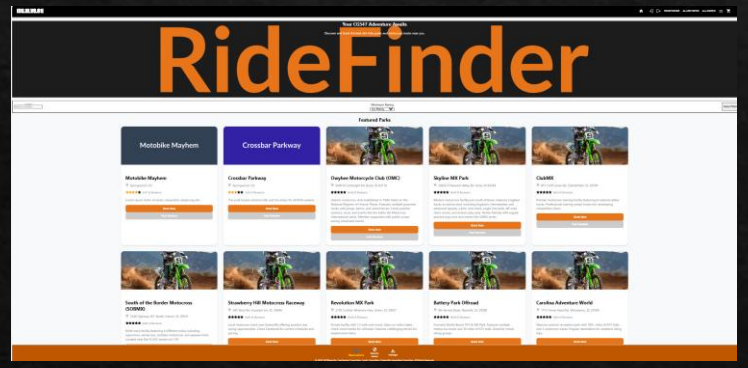
Dev Hosting(CTS Business Pro on G.Cloud)

DEV->1 VM 8GBx2Cores, Linux XFCE VDI

Production Hosts

Azure App

CG RideFinder Requires Northbound and Southbound Interfaces which can be simple architecturally from the CG perspective.... (Get Parks) & (PostCart)



Browser Local Storage
Keeps CG
Cart Details & Icon Updates

CGPARKS API

CGCART API

GCParksDTO

GCCartDTO

RideFinder without Auth Really only has two main functions with simulated payments....which can be multiplexed in with the Cart Transaction.

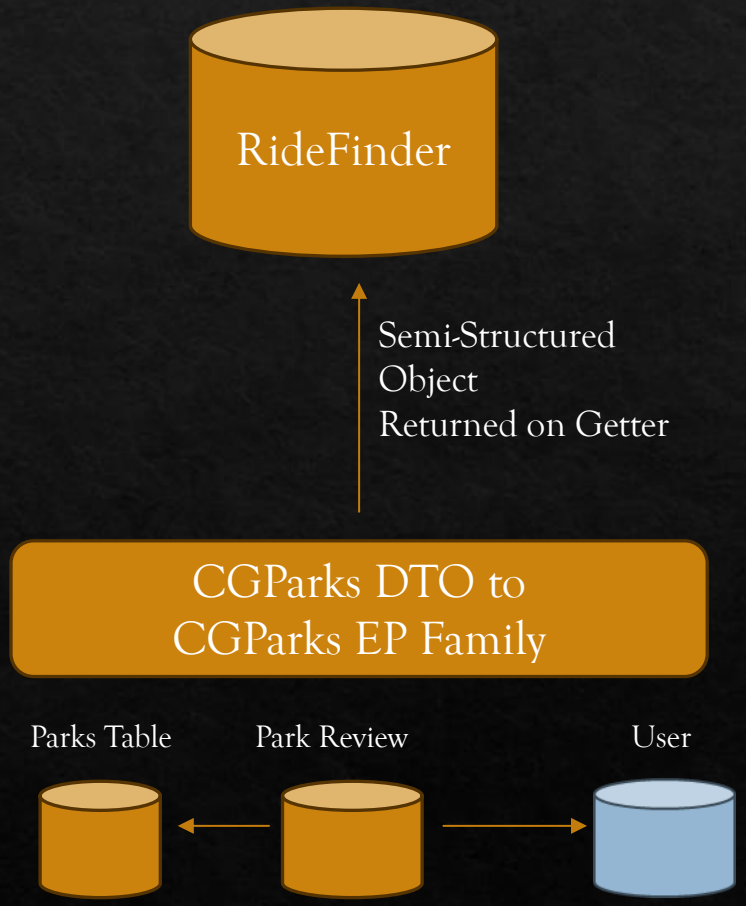
Allowing the Whole Transaction to Be Approved or Denied in the BackOffice with a Success Allowing the Cart to Be Cleared. But This is a Mask.... As Cart Spans 5 Tables, and Parks 3+

Ride Finder Architectural Decisions.....

- NO OPTIONS ON NORTHBOUND INTERFACE
- The Northbound Interface is fully defined, and the application literally doesn't work unless the JSON structure matches exactly what is in the JSON examples.
  - To change the Northbound Interface would require a near re-write of the whole application.
  - CapGemeni told us in the last week that we had to consider CG the owner of RideFinder Application and that we should not change the UI without permission.
  - We decided not to make any changes to the main selector screen for the NB interface, but we did disable the filters so we could see if the application was receiving the right data.
  - We did however have to rewrite the ParksService Controller to PULL FROM AN API RATHER THAN A JSON..... Since we were hosting an AZURE API... this took about 100 lines of code or so.... But its possible we could have just exported a json to the Ridefinder Disk running the API, and rewrite the APP to read from disk. We choose the easier option in-line with the project requirements.
  - Only Minor Issue is to SEED the Default Parks OR NOT? We choose to Multiplex local JSONS with CGPARKS DTO, but this makes an error on creating a Review.
- SOUTHBOUND INTERFACES CAN BE DONE IN BULK OR IN PART LIKE PARKS APP
  - Architecturally Attempting to Post AN ENTIRE CG CART unmodified is the easiest pathway for UI Bypass.
    - For this to work... you just read Local Storage to get the Cart, and send it to POST with some return message for success.
    - This requires almost no major changes to the UI.
  - OPTION1 – Bulk Post with Everything happening on the BackOffice.
  - OPTION2 – Payments as an IoT application and everything else happening on the BackOffice. This requires a DTO Southbound.
  - OPTION3 – A Complex Multi-API Thread where a thick client is hitting multiple APIs all over the Back Office. This doesn't require a DTO South.
  - OPTION4 – Hybrid – Post the Full Cart, Or A Cart Item, And Use Various APIs for Validation, Auth, Payment Recording, ETC.
  - DBOPTION-A – A Fully Semi-Structured Backoffice From End To End
  - DBOPTION-B – A Complex Distributed SQL Database Architecture with DTOs simulating Semistructured Data facing the UI.

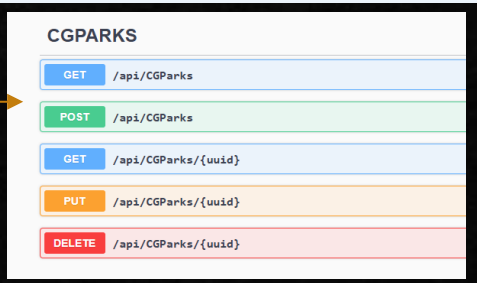
- The Southbound Interface Carries a Triple Black Diamond Rating due to a DTO, and Parsing Complexity.
- The Northbound Interface is a Single Black Diamond due to the DTO.

Northbound -> Data Architecture Tree View A Complex, Unusual UI Requirement

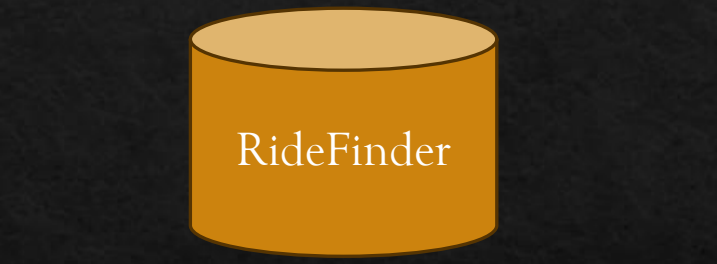



```
[
  {
    "id": "string",
    "parkName": "string",
    "location": "string",
    "description": "string",
    "adultPrice": 0,
    "childPrice": 0,
    "imageUrl": "string",
    "reviews": [
      {
        "author": {
          "id": "string",
          "displayName": "string",
          "fullName": "string",
          "dateOfBirth": "string"
        },
        "rating": 0,
        "dateWritten": "string",
        "dateVisited": "string",
        "review": "string"
      }
    ]
  }
]
```

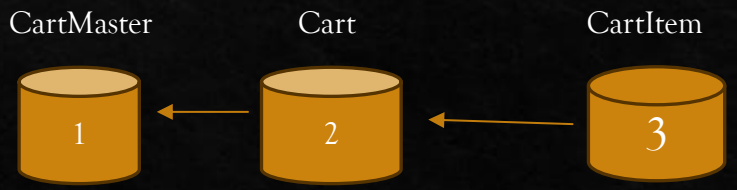

Technically only Get
EP is needed... Others
Extraneous

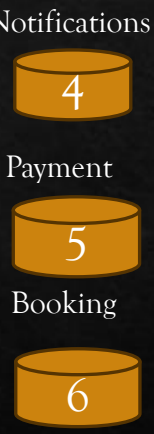


Southbound -> Option1 -> Data Architecture Tree View Decomposition

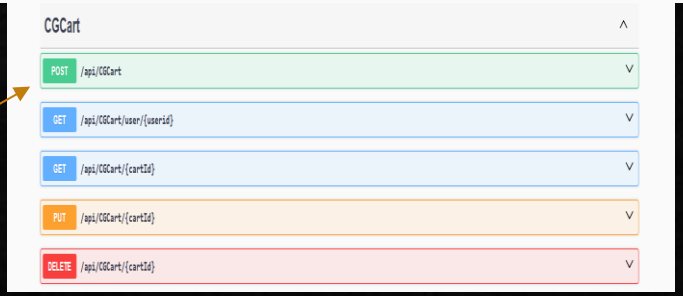


SemiStructured
Object
Sent on Post





Technically Only
Post is Needed



CHANGES POST RAPID DEVELOPMENT -> CG Ridefinder App

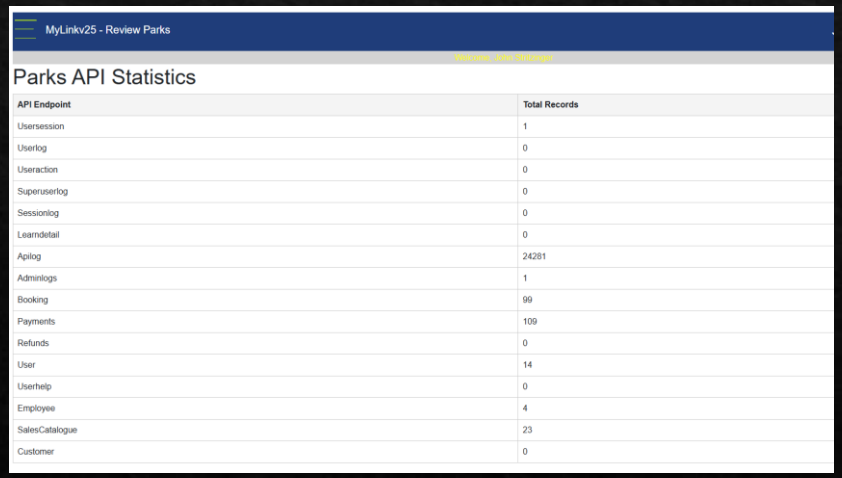
- In Phase0 to Phase1 -> We decomposed the final view SANS the CG UI, and tried to build a working UI like we did in both the Fall of 2024, and Spring of 2025. Adding an additional UI was arbitrary Department decision.
- The Parks System Supports Both Dirtbikes, and All Other Park Functions inclusive of more park features, a better sales interface, and better visualizations, more management features, and Auth controllers.

- BUT THAT DECISION WAS MADE SO... WE MOVED ON....

- The Ridefinder App had just a few differences in schema than what we envisioned including:
  - Park Default Pricing
  - Park GUID Indexing (Which was a big systemwide change)
  - Multiplexed Park Reviews

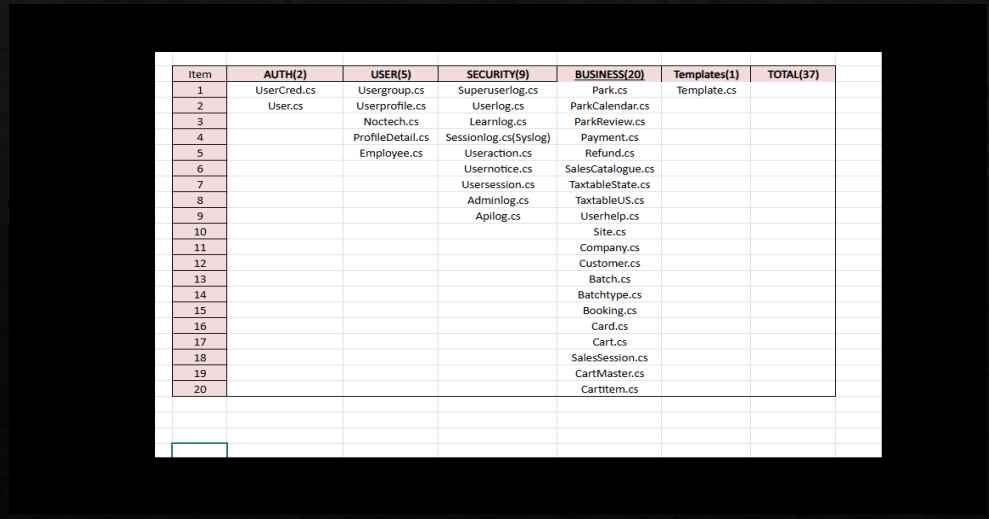
Scope of Project - Effort

- 37 Table Environment To Support A Professional Environment… BUT… 6 Tables (Logs, CartSemi, ParkSemi, User, Booking, and Payment is a possible minimal effort after receiving CG UI… IE had you waited or had no code).
- > 24K API CALLS SINCE 11/1 -> Massive Amount of testing
- > 200 End to End Transactions Simulated and Posted
- > UI NORMALIZATION (RIDEFINDER VS PARKS). > 2 DAYS….WITH DTOs and New Endpoints.


MyLinkv25 - Review Parks

Parks API Statistics

API Endpoint	Total Records
Usersession	1
Userlog	0
Useraction	0
Superuserlog	0
Sessionlog	0
Learndetail	0
Apilog	24281
Adminlogs	1
Booking	99
Payments	109
Refunds	0
User	14
Userhelp	0
Employee	4
SalesCatalogue	23
Customer	0



Item	AUTH(2)	USER(5)	SECURITY(9)	BUSINESS(20)	Templates(1)	TOTAL(37)
1	UserCred.cs	Usergroup.cs	Superuserlog.cs	Park.cs	Template.cs	
2	User.cs	Userprofile.cs	Userlog.cs	ParkCalendar.cs		
3		Noctech.cs	Learnlog.cs	ParkReview.cs		
4		ProfileDetail.cs	Sessionlog.cs(Syslog)	Payment.cs		
5		Employee.cs	Useraction.cs	Refund.cs		
6			Usernotice.cs	SalesCatalogue.cs		
7			Usersession.cs	TaxtableState.cs		
8			Adminlog.cs	TaxtableUS.cs		
9			Apilog.cs	Userhelp.cs		
10				Site.cs		
11				Company.cs		
12				Customer.cs		
13				Batch.cs		
14				Batchtype.cs		
15				Booking.cs		
16				Card.cs		
17				Cart.cs		
18				SalesSession.cs		
19				CartMaster.cs		
20				Cartitem.cs		

User Stories... Recycled from 590Team1

- Flows of Portal Actions were all designed by 590Team1 (Sambit, John, Pravani, Carolina) include Admin Bounces based on Role.
- G/A PLL(Page Level Locking) has been used across all Generations of Projects.
- Has reduced the complexity of AzureDevOps Recording Dramatically.

Functional View?

- Project/Scope of Work (Venkata, John, Lily, Jackson et all)…
- ReactUI (Lily, John, Jackson Spanning 18 months).
- IonicUI (John Only)
- AzureServices(John(File,DB, Hosting,Functions),Carolina(Blob), Colin(ServiceBus), Sambit OCR, Kaleb Auth).
- DB Architecture (John at 99%, Jackson at 1%) for all Teams and Generations.
- EP Design (John at ~97%, Caleb at 3%)
- HTML/JS Front-End(John Bulk Efforts, Pravani CSS Template(Features, Feature, Auth Pages))
- JSS Has Rewritten Simpler Version of All Auth Pages without encryption, and using Social, and 3rd Party CMS.
- JSS Has Ported Old Code to 4 DBMS Platforms including SQL Azure left intact. (590LOGIN Unaltered still works).

Seeding

- Default Parks Need to Be Added to Parks DB in SQLITE to write reviews for them.
- NC Parks
- VA Parks
- PA Parks
- MI Parks
- Are all available via CLI Interfaces.

Code/Review -> Questions?

- Major Rewrites UI
  - RideFinder (Cart/Park Services) Moved to API from JSON.
  - RideFinder (Payment Service) Had More or Less no Working Code.
- BackOffice
  - RideFinder requires ParksDTO, and CartDTO, Services, and at least 2 Endpoints.

CG Baseline vs Industry Standard UI Requirements

- CG Baseline (A Proof of Concept)
  - CG UI As-Is relies on Semi-Structured Park Data as Inputs to the Service Architecture. (GET)
  - CG UI As-Is relies on the Post and Follow-On Actions Following Sale. (PUT)
  - In general only 2 EP's are Needed for the CG UI but they are being driven by complex services which map several tables data into JSON Formatted API Output.
- IS Baseline in Excess of the CG Offer
  - Auth Controller, User Services, Management Reporting, User Reviews, UI Templates & Icons.

DB Requirements For CG SOW

- CG Needs (Parks, Payments, Bookings, Cart, CartItems, User)
- GA Needs (30+ Table Environment -> Major Upgrades-> Security, Product, User)
- EPS
  - CG Endpoints to Drive their UI -> Translations to the Parks DBMS.
  - GA Parks Endpoints to Drive 3 UIs -> Full Featured Database Backoffice
  - Speciality Controller for Backoffice Tasks (Park Averages, Seeding, Utilities).

Backup Slides

Details -> Code Analysis, Graduate Extensions, Process Flows, Repo Views

Project Executive Overview

Program Operations

Highlights

- We have 1 Master Repo with all the Code
  - It contains five sub projects (API, CGUI, PARKSAPP, PARKSUI, PARKSCLI)
  - It contains project resources and sprint documents.
  - We have updated and shared this summary repo to all CG Consultants.
- Azure however has unique build requirements
  - It doesn't handle subdirectories very well even though the forms look like they will work.
  - We therefore have 5 decomposed Github Projects to support the massive code base. These are not shared, as intermediate processes are not material. But they are relevant for workprojects.
  - We added a New Azure Devops Repo for CG2 UI.
- REQUIRED CODE
  - Parks API Builds
    - The Parks API has gone through 16 Major Revisions.
  - CGUI
    - The CG UI has gone through 6 Major Revisions since Sprint1.
- LOGGED ACTIONS
  - We have logged more than 2500 Endpoint Calls, and Run more than 250 End to End Transactions through the system.
  - This is in excess of debugging each and every screen multiple times to check every post, and get from the UI to the BackOffice.
- CHANGES
  - After an initial port of 590 Code, We Built a Full UI for Parks which does everything that a Modern Ski Resort, and KOA might do.
  - After CG provided a UI, we upgraded with Code Jackson and I built in 547 with Material UI styling and Icons.
  - We have made hundreds of changes on this platform nearly every day for many months.

Historical Comparison & Lessons

- 547Team1 – Fall 2024 (HARD, but a large team) – 5 People
  - Movie Project had a Movies Database, A Cart, and a Manager…. But required user services. It was significantly harder in practice because showtimes were different per Movie Theatre, and because Movie Theatres necessarily imply a branch environment. It was an extreme coding exercise but with 5 professional developers. (Challenges -> React Axios didn't work in production, and Database Changes were massive so that in practice non of the intermediate builds worked either….)
- 590Team1 – Spring 2025(UI Easy, Portal Hard, And Auth Hard) – 6 People
  - Team had to build automation for Professional Certifications and Automate User Auth tasks however only a single branch was considered for a single company. User Profile in a Decomposed Structure is hard. We could have done better with a single table.
- 547Team5 – Fall 2025 (UI Very Easy, DB Construct Very Hard) – 1 CG Consultant, 1 GS, 5 Matrixed.
  - CapGemeni gave us an extreme Programming challenge on the backend. With a Semi structured Model this was probably on the surface an easy project. However teams that implemented with two semistructured tables would have built a nearless worthless project. We decided we wanted to build a cart and eCommerce Engine we could keep forever.
  - Teams choice of SQLITE was helpful at first, but is a hinderance when it comes to notification and alerting.
  - Integrating Auth from my old team was seemless as I used the same table architecture.
  - However I had to rebuild Userprofile, and remap all the Javascript to clean up what ended up as a mess.
  - IN GENERAL I BELIEVED WHEN I STARTED AND BEFORE WE RECEIVED CG CODE, THAT I HAD TO REUSE OLD CODE AND BUILD MY OWN UI TO BE SUCCESSFUL. I PROBABLY WOULD HAVE MADE DIFFERENT DECISIONS HAD WE HAD THE UI INITIALLY.

Requirements

- CG Process For This Assignment is Streamlined
  - CG UI -> Receives Semi-structured Data by Default* from Parks, and Park Review Tables.
  - CG UI -> Takes Basic Financials from Park Inventory, and Creates Shell Records in LocalStorage. Requires these CartDetails to be posted back to the System.
  - API -> Sends Semi-Structured Data(Getter) from Parks, and Park Review Tables.
  - API -> Receives Semi-Structured Data (Post) from Cart, User, Park By Design.
- The System as designed for this assignment without capacity checks requires data to be assembled and disassembled OR saved like for like in the Parks SQL Backend.
- The system is missing User Functions, All Security Features as designed but students in nearly every team project in this university have to build or use COTS tools for these services.
- We decided on a low security model to start, and have upgraded the security dramatically for Phase2 across the platform.
- Teams that waited for the CGUI, would have been rewarded with perhaps a 2 table semi-structured database which worked on the exact form the system created. But the resulting platform would be nearly worthless, and without any managerial features.

The Process In Review

- Sprint 0 Review
- Team 2 Decided on SQL Lite and Flat File.
- This was helpful for Sprint0 and completing an initial project plan in time.
- Sprint 1 Review
- We decided to do a rapid development phase to lay out the entire project from end to end with a professional plan in mind.
- Rapid Development resulted in a nearly formed back office without triggers, or DB constraints by 11/10 and two UI programs which we presented during Sprint1.
- During Sprint1, and after Sprint0 we received the CG UI. It had several fields which needed to be mapped into SQL including default pricing per park. It also used displaynames for the User record which needed to be added.
- DB Changes for the CG UI were complete by 12/1 (Relatively short).
- Massive changes from 12/1 to now on the UI including 10+ new pages and integration of the CG Semistructured Datamodel which was a bit audacious and varied greatly from our Rapid Development approach.
- Sprint2 Readiness
- We have a solid base of code posted which allows us to demonstrate all end to end functionality without capacity checks, and park level calendars. We have 3 Full UIs working to the same API.
- Our API is professional Grade and can support any enterprise. It includes all the functions of all three teams, and the CG UI. It is an industrial platform built over 18 months.

Contributions By Major Area of Effort

- GS(5)Overall - > 1 FTE(John), and 6 USC Matrixed Resources, 1CG UI Consultant
- React UI -> (Lily, John, and Matrixed Jackson(Fall2024).
  - Lily Excellent Base Code to Working Level.
  - Added Jacksons Material UI Menu System He Built last fall.
  - Used Jacksons Menu Layout Initially, but Added a Menu Bar which is new, and upgraded icons from last fall.
  - Added My Bottom Bar and Managers from Fall of 2024.
  - Morphed Colors and Templates to match Customer requirements.
  - 3FTES in practice on UI.
- Parks UI ->(John, Pravani, Sambit, Carolina) ->Ghost Portal
- Parks API ->
  - Phase 0 ->590Team Auth Controller & Users -> Upgraded and Ported(JS). (Caleb, Colin, John)
  - Phase I -> John All New.
  - Phase II -> Overlay with CG UI, and DTO requirements to migrate from SQL to SemiStructured.
- Parks Security ->
  - John All New. Logs for Everything Server Side AND Client Side. New SecurityHome Page.

What I learned this semester.....

1. Slaying the Dragon
   A. From Angry to Satisfied
      - Honestly... my team from last fall gave a demo which I didn't think worked....
      - It would have failed change control at my old companies.
      - It had more than 30 migrations on the database and none of them ever worked on my environment.
      - I had to work around their model with Direct to Database Microsoft Grid Objects, CLIs and tools not contemplated by the core team.
      - I didn't think the Pomelo drivers picked by my team were technically sufficient and gave weird messages with complex joins in the logs.
      - A Massive React Project was abandonded last year with a working and in production cart with Stripe integration.
      - My team deleted my whole production server... and well I believed it was criminal in nature.
      - I hadn't come to grips with my teams project until almost Feb1st 2025. It was in fact a violent experience.
   B. Production Ready this year.....As the Tech lead of my old team... I was able to fix all of these things this year, and using FETCH vs Axios fixed most of the technical issues.
   C. Building EP Templates saved my career at the University. I could not have done anything in any of my research without them.
2. Speed-> I can build Per Table EPs in less than an hour, last year they took more than a week to carve custom code from a C# code base I was still learning with a 1200 page book as mostly the only reference.
3. Understanding of REST, AJAX, JQUERY, COPILOT, GITHUB MUCH better than previous two classes.
4. Understanding of the Cost Metrics for Azure have made me much more comfortable working with Microsoft and explaining why we had cost overruns in certain areas.
5. Lily's React Framework has been very helpful in mirroring Structured Programming for the Mobile Space inline with our back office. I am getting to be a much more competent react developer.
6. All of the code samples provided by CapGemeni have been very helpful. These projects have been brilliantly orchestrated. Well done.