Group 4

Campus Compass

https://github.com/patelanay/CampusCompass

Anay Patel, Ignatius Martin, William Chi, Giovanni Sanchez

Section 1:

Description

CampusCompass is a web-based calendar and scheduling assistant for UF students that centralizes classes, exams, and campus events into one interface, with secure login handled by Google OAuth using students' UF-linked Google accounts. It aims to reduce missed deadlines and fragmented schedules by synchronizing academic and personal commitments in a single systemIt allows a UF student to log in with Google, view a personalized calendar, and manage tasks tied to course schedules and campus activities.

Solution to the Challenge Statement

The core challenge is that UF students juggle classes, assignments, exams, and events across multiple platforms (Canvas, email, personal calendars), making it easy to overlook important deadlines. CampusCompass addresses this by consolidating time-management information into one tool, enforcing authenticated access via Google OAuth so each user's schedule is securely tied to their UF account.

Features and Functionality

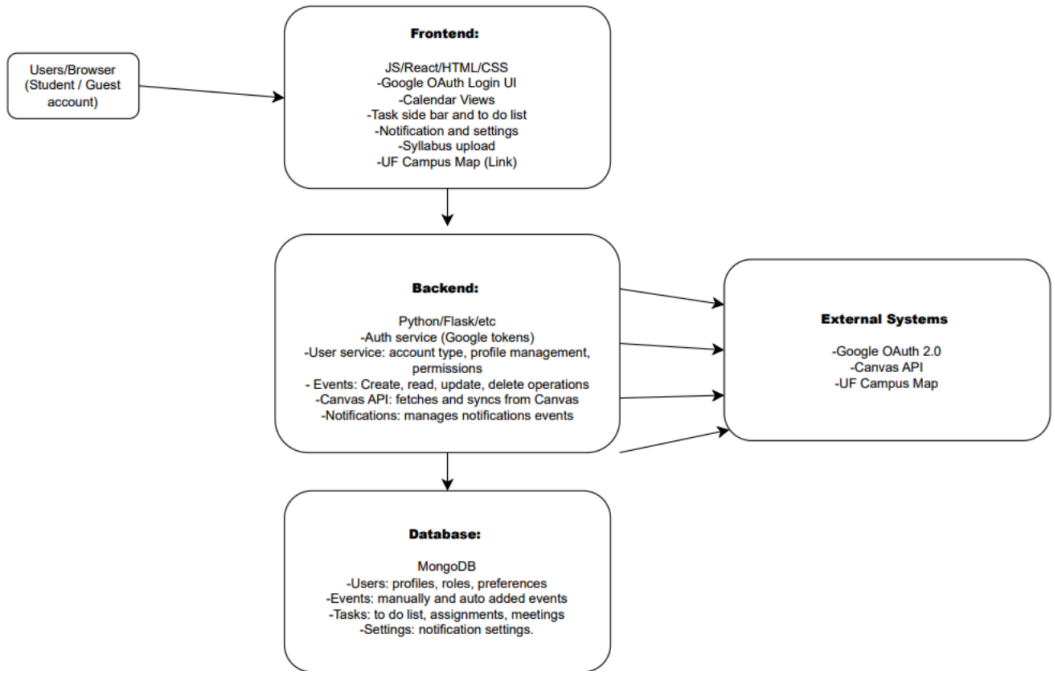
CampusCompass provides secure Google OAuth login so UF students can authenticate using their UF-linked Google accounts, then access a personalized dashboard that summarizes today's and upcoming items such as classes, exams, assignments, and campus events in one place. After logging in, students can view their schedules in calendar views while being capable of creating, editing, and deleting custom events that are stored in the backend and associated with specific courses or tags to keep academic and personal items together. Moreover, freshmen may find it difficult to locate their classes given the University of Florida's large campus, so CampusCompass offers a direct link to UF Campus Map.

System Models

Frontend (Presentation) Layer will be implemented using React, JavaScript, HTML, CSS, etc. This layer is responsible for the user interface and graphics. This layer is also responsible for handling the logins and authentication, with features such as calendar view, task bar, hyperlinks to the UF map, etc.

Backend (Application) Layer will be built with Python, etc. This layer manages the background application logic including the authentication through Google OAuth. This will also handle the guest vs student account types and the integration with the API. This layer handles the response exchange between the frontend and backend.

Database (Data) Layer will be implemented with MongoDB which stores key information such as user profiles, scheduled events, tasks, rules, etc.



Section 2:

Code Management:

For CampusCompass, code management relied on Git and GitHub with a simple feature-branch workflow to keep the main branch stable and make collaboration clear. The team kept a protected main branch for production-ready code, created separate branches for new features or fixes, committed changes frequently with descriptive messages, then opened pull requests so teammates could review and merge only tested, working code back into the master branch. Everyone regularly pulled from the remote repository to stay up to date, resolved merge conflicts on their feature branches, and deleted merged branches to keep the repo clean, which aligns with standard version control best practices for small student teams.

Project Management:

For CampusCompass, project management followed a Agile-style approach tailored to a small student team: the group broke the work into clear milestones (such as “set up Google OAuth,” “build calendar view,” and “integrate UF map links”), assigned owners, and tracked progress week by week so that scope stayed realistic alongside classes. A

Kanban-style board listed all user stories and bugs; team members moved cards across columns as work advanced, which gave everyone visibility into who was doing what and helped avoid duplicated effort. The team held short check-ins during work sessions or over chat to review status, reprioritize tasks, and unblock issues, then used informal retrospectives after each milestone to discuss what went well or needed improvement, aligning with common Scrum-inspired practices for student software projects.

Test Plan:

For CampusCompass, the test plan focuses on verifying core functionality (Google OAuth login, calendar access, event management, and UF map) from an end-user perspective and ensuring the system is stable enough for everyday student use. Testing is organized around a small set of high-priority scenarios: successful and failed Google logins, session and access control to protected pages, CRUD operations for calendar events, and correct creation of UF Campus Map link so freshmen can reliably navigate to their classes.

The scope of testing includes functional testing of all user-facing flows in a modern browser, basic usability checks (clear error messages, intuitive navigation), and light regression testing whenever new features are added, while non-critical areas like performance under heavy load are considered out of scope for this course project. The team executes manual tests on a shared staging/development environment using a defined set of test cases (such as the five Google OAuth-related cases you wrote), records pass/fail status, and logs defects as GitHub issues so they can be prioritized and fixed before the next milestone. Entry criteria for a test cycle include completion of feature implementation and code review, and exit criteria require that all high-priority test cases pass with no unresolved critical bugs, aligning with standard test plan structure for small web applications.

Section 3

Technical Details:

The client side is built as a browser-based interface using HTML, CSS, and TypeScript to render login views and calendar pages, while the server side exposes routes for authentication, session management, and calendar operations. Communication between client and server occurs over HTTPS, and user-specific data such as events and location metadata is persisted in a MongoDB database, so each student's schedule and navigation links are maintained across sessions.

Authentication is handled using Google OAuth 2.0 with the web-server authorization flow. When a user selects "Login with Google," the application redirects the browser to

Google's authorization endpoint, where the user authenticates and grants consent. Google then returns an authorization code to a designated callback route on the server, which exchanges the code for access and/or ID tokens. The server establishes a secure session (for example, via an HTTP-only session cookie) tied to the authenticated user, and all protected routes such as the dashboard and calendar views verify the session before returning data. Unauthenticated requests to these routes are redirected back to the login entry point, ensuring that calendar data is only available to logged-in UF students.

The core scheduling logic centers around a calendar data model stored in MongoDB collections. Each event document includes attributes such as title, start and end times, course identifier, event type (e.g., lecture, exam, assignment, personal), and an optional location code, along with a reference to the owning user. Server endpoints implement create, read, update, and delete operations by querying and updating these MongoDB collections, typically using indexed fields like user ID and date/time for efficient retrieval. On the client side, events returned from MongoDB are rendered in different views (such as weekly and monthly), with options to filter by course or category, allowing students to manage academic and personal items in one unified interface.

A distinct technical feature of CampusCompass is its integration with the UF Campus Map to support physical navigation on a large campus. Each class or event that has a physical classroom is associated with a building or room code; CampusCompass extracts the first three letters of this code (for example, "CAR") and uses that prefix to look up the corresponding campus-map URL for the building. This mapping from building prefixes to URLs can be stored either as a configuration collection in MongoDB or as a static mapping in the server code. When events are displayed to the user, the location field is rendered as a clickable hyperlink that opens the appropriate page on the UF Campus Map in a new tab, allowing freshmen to quickly locate their classrooms directly from their schedule.

Installation Instructions:

1. Navigate to backend folder, install requirements, and run main.py
   cd backend; pip install -r requirements.txt; python main.py
2. Navigate to frontend folder and run npm
   cd frontend; npm install; npm run dev
3. Open in browser the localhost link after you npm run dev

Login and Access Credentials and API Keys:

We utilize 3 .env files.

1. Parent folder .env
   #MongoDB
   MONGODB_URI="mongodb+srv://wchi:cnvsCompass321@c-compass.yh

tmbod.mongodb.net/?retryWrites=true&w=majority&appName=c-compass
"
#Google VITE_GOOGLE_CLIENT_ID =
1065291463661-l2106f9imanqem8usqppof8fbseeojbf.apps.googleusercon
tent.com GOOGLE_CLIENT_SECRET =
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

2. Backend folder .env

```
# Google OAuth Configuration
GOOGLE_CLIENT_ID=1065291463661-l2106f9imanqem8usqppof8fbsee
ojbf.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=GOCSPX-cikspIB-oqCAL7UnJ3JyYAicaes
V

# MongoDB Configuration (if using)
MONGODB_URL=mongodb://localhost:27017
DATABASE_NAME=campus_compass
MONGODB_URI="mongodb+srv://wchi:cnvsCompass321@c-compass.yh
tmbod.mongodb.net/?retryWrites=true&w=majority&appName=c-compass
"
```

3. Frontend folder .env

```
# Google OAuth Client ID
VITE_GOOGLE_CLIENT_ID=1065291463661-l2106f9imanqem8usqppof8
fbseeojbf.apps.googleusercontent.com
```

Section 4

Risk Management Plan:

We utilize a plan that involves identifying risks during early planning and revisiting them at each milestone with a focus on technical, security, schedule, and usability issues that could affect UF students' ability to log in, view calendars, or use campus navigation. Each risk is described, rated for likelihood and impact, and assigned an owner responsible for monitoring it and executing mitigation actions when necessary, with the risks tracked in a simple shared document or issue tracker that serves as a small risk register. Key risks include Google OAuth integration problems, which are mitigated by following Google's OAuth guidelines, testing the login flow in development, and keeping client ID, secret, and redirect URIs clearly documented in environment configuration. Security risks arising from incorrect handling of OAuth tokens or sessions are reduced by using secure session cookies, validating tokens server-side, restricting sensitive

routes to authenticated users, and regularly reviewing OAuth scopes and test credentials. Data integrity and availability risks for MongoDB are addressed by storing connection details in environment variables, testing connectivity before deployment, using simple documented schemas, and maintaining basic backup or export procedures for development data. Schedule and scope risks due to semester time constraints are mitigated by prioritizing core features, de-scoping lower-priority enhancements if needed, and using a visible task board to monitor progress and adjust workload early. Finally, usability and navigation risks, such as confusing UI or incorrect UF Campus Map links, are managed by informal user testing with classmates, manually verifying classroom codes and their generated links, and fixing inconsistent mappings before final deployment, with the team briefly reviewing and updating all risks at major milestones.

Software Quality Attributes and Explanations:

Usability:
CampusCompass is made for UF students, mainly freshman, so simplicity is used for the design to convey information effectively. The direct links to the UF Campus map allows for less confusion on where to go for classes without looking classes up individually.
Reliability:
We use well-supported systems such as MongoDB, Google OAuth, and FastAPI/Node frontend, which ensure system reliability. Using a Google login and creating a calendar page were tested manually throughout development of the project.
Security:
Security is built around Google OAuth 2.0, meaning that rather than storing passwords locally, authentication is handled by Google. Sensitive credentials, including the MongoDB URI, the client IDs, and secrets, remain in .env files and are never committed to GitHub.
Maintainability:
The frontend has been refactored for readability, with much cleaner component organization; the route definitions of the backend are clear and modular. Due to the separation of frontend, backend, and database logic, updates or extensions of the project are fairly easy.
Performance:
Communication between the client and server is via lightweight JSON over HTTPS. For this course, high-volume performance testing was not necessary, but the structure here easily allows for future optimization if necessary.
Scalability:

CampusCompass can scale beyond the current scope without major redesign. MongoDB is able to scale in order to store increasingly large amounts of user data, while the API-based backend allows additional frontend features. The UF Map can easily be swapped or extended to other campuses.

References

https://docs.python.org/3/library/datetime.html
https://docs.python.org/3/library/uuid.html
https://docs.python.org/3/library/typing.html
https://fastapi.tiangolo.com/tutorial/path-operation-functions/
https://www.w3.org/TR/NOTE-datetime