Udemy Upgrade Videos

Due: See online for due date/time **Collaboration:** Group Assignment **GenAI:** Limited (see below)

Udemy Apps to Upgrade

After completing the necessary Udemy Progress Video assignment in the Udemy course, you will be challenged to make upgrades to 4 specific apps that you build via the tutorials, as seen below:

1. **Quiz App**
   a. Covers Section 3
2. **Expenses App**
   a. Covers Section 5-6
3. **Meals App**
   a. Covers Sections 8-10
4. **Shopping App**
   a. Covers Sections 11-12

NOTE: Each of the 4 apps above represents a different assignment over the course of several weeks. Thus, the instructions below should be carried out 4 times throughout the semester.

Submission Instructions

What to Submit

Submit the following:

1. A **PPTX/PDF file** containing the flutter upgrade summary slide template which you modified for this upgrade
2. A **7+ minute video** (non-private YouTube link or direct upload) with the elements as described and in order in the “What to Create (Video Assignment)” section below

What to Do

Upgrade Quantity

After completing the appropriate app tutorial from the Flutter & Dart - The Complete Guide Udemy course, complete several upgrades, based on your team size:

- **Individual:** At least 2 (two) upgrades
- **Group of 2:** At least 3 (three) upgrades
- **Group of 3 (rare):** At least 5 (five) upgrades

NOTE: Each team member must invest significant time as the "lead developer" on at least 1 upgrade.

Upgrade Quality & Challenge

Challenge Ladder

Each upgrade must be selected from the **12-Level Upgrade Challenge Ladder** below. The ladder defines examples, effort goals, and challenge rationales to help you select meaningful upgrades that are appropriate for each app. As the semester progresses, you are required to attempt more challenging upgrades from higher levels.

Upgrade Rules by Assignment:

- **Upgrade 1 (Quiz App):** Levels **1–12** allowed.
- **Upgrade 2 (Expenses App):** Levels **4–12** required.
- **Upgrade 3 (Meals App):** Levels **7–12** required.
- **Upgrade 4 (Shopping App):** Levels **7–12** required.

Each upgrade assignment should require approximately **3–6 hours** total per student (so a team of two students should expect to spend about 6-12 hours, in total, to complete their 3 upgrades on each upgrade assignment); if you are spending far more time, your upgrades may be too complex (though going above and beyond is welcome).

The 12-Level Upgrade Challenge Ladder

App	Level	Examples	Effort Goal	Challenge Rationale
U1	**1 – Styling Touches**	Change multiple colors, fonts, or themes; consistent theming across widgets	30–60 min	Teaches you to think about app-wide consistency beyond a single widget
U1	**2 – Minor UI Enhancements**	Add icons, gradients, or simple animations; restructure layouts with Rows/Columns/Flex	1 hr	Pushes you to improve layout hierarchy and polish
U1	**3 – Simple Widget Swap/Add**	Add Slider, Stepper, or Switch; create small reusable widget	1–1.5 hrs	Expands your widget knowledge and modular design skills
U1–U2	**4 – Basic State Use**	Toggles, counters, visibility changes, user input tracking	1.5–2 hrs	First step into interactivity — managing widget state
U1–U2	**5 – Basic Package Integration**	Star rating, carousel, emoji picker from pub.dev	2 hrs	Practice using external docs & packages
U1–U2	**6 – Forms & Input Validation**	Forms with validation and error messages	2–2.5 hrs	Teaches good UX around input flows
U1–U4	**7 – Multi-Screen Navigation**	New detail/settings/profile screen; pass data between screens	2.5–3 hrs	Expands app into multi-page workflows
U1–U4	**8 – Intermediate Package Integration**	Charting, maps, or image picker with preview	3 hrs	Connects external package behavior with your own state
U1–U4	**9 – Persistent Local State**	Save data in Hive/SQLite/SharedPreferences	3–3.5 hrs	Adds persistence, async data, lifecycle thinking
U1–U4	**10 – Complex State Management**	Use Provider for shared state across screens	3.5–4 hrs	Stretches you into app architecture & refactoring
U1–U4	**11 – Cloud/Firebase Features**	Firestore queries; Firebase Auth; Firebase Storage uploads; Some other new GCP feature we haven't used	4–5 hrs	Real-world async + client/server thinking
U1–U4	**12 – Major Feature Extension**	Workflow combining multiple skills (e.g., product upload → image → Firestore)	5–6 hrs	Integrates multiple skills into a polished, mini-feature set

Upgrade Purpose as True Features

Each upgrade should be presented as a **self-contained, meaningful feature** with a clear purpose. Don't just add a "random package" — your upgrade should add tangible value that a client would appreciate.

App-Specific Examples

The following list provides ideas for how you might achieve a specific ladder on a specific app; you are encouraged to do something creative that is of interest to you, so long as it falls into one of the **Challenge Ladder** categories:

1. **Quiz App (Upgrade 1: Levels 1–12):**
   - **L1 (Styling):** Apply custom color scheme and consistent text style across all questions.
   - **L2 (Minor UI Enhancements):** Replace the vertical answer-button stack with a responsive Wrap/Grid to prevent overflow on small screens.
   - **L3 (Simple Widget):** Add a progress bar to show quiz completion.
   - **L4 (State):** Track and display the current score live during the quiz.
   - **L7 (Navigation):** Add a "High Scores" screen.
   - **L9 (Persistence):** Save scores locally in SharedPreferences.
   - **L11 (Firebase):** Sync scores to Firestore for a class leaderboard.
2. **Expenses App (Upgrade 2: Levels 4–12):**
   - **L4 (State):** Add a toggle to filter expenses (e.g., "show last 7 days only").
   - **L5 (Package):** Use a chart package to display expenses by category.
   - **L6 (Forms):** Add input validation (e.g., disallow negative expenses).
   - **L8 (Package Integration):** Add Google Maps to attach a location to expenses.
   - **L9 (Persistence**): Store expenses in SQLite/Hive for offline use.
   - **L11 (Firebase):** Sync expenses to Firestore with user authentication.
3. **Meals App (Upgrade 3: Levels 7–12):**
   - **L7 (Navigation):** Create a "Favorites" screen to view saved meals.
   - **L8 (Package Integration):** Add image-picker for users to upload their own meal photos.
   - **L9 (Persistence):** Save favorite meals locally in SQLite.
   - **L10 (Complex State):** Use Provider to manage favorites across screens.
   - **L12 (Extension):** Add Firebase Auth so users have own meal plans stored in Firestore.
4. **Shopping App (Upgrade 4: Levels 7–12):**
   - **L9 (Persistence):** Persist shopping cart contents across app restarts.
   - **L10 (Complex State):** Use Provider to handle cart/favorites across multiple screens.
   - **L11 (Firebase):** Sync shopping cart to Firestore so it follows the user across devices.
   - **L12 (Extension):** Build checkout workflow: login→add items→upload order to Firestore.

General Upgrade Resources

To help with possibilities of things you can use to implement ideas:

- search through the Flutter material component widget libraries and properly implement new Flutter widgets that have not yet been seen/used
- search through pub.dev for new Flutter/Dart packages to import and utilize in your project

NOTE: Keep in mind that importing new widgets or packages into your own app comes at all different levels of difficulty and the instructor is well-aware that importing and using some packages may be very simple, so package imports are often included as part of a feature upgrade (e.g., a star rating package *may be used to enhance the overall feature being presented, which is a new rating page that had not existed)*

What to Create (Video Assignment)

When finished, create a **7+ minute screen-share video** and submit it as a **public or unlisted** YouTube link (or raw video, if you absolutely must, but this is not preferred) to Blackboard. In the video, you will need to include the following elements IN THIS ORDER (not doing so will result in a penalty):

1. **(~1 min) Slides (Intro & Flutter Upgrade Summary):** Show the updated and completed intro and flutter upgrade summary slides, talking through the upgrade details and work breakdown for the features you are about to present
    a. You MUST use the provided template and fill out a row for each unique feature you are claiming as a stand-alone feature
    b. Each column of the rows you are claiming as a standalone feature must be completed
2. **(~1 min) Flutter Demo:** Demonstrate ALL upgrades IN THE ORDER in which they are listed in the overview slides in a live MOBILE-APP simulator (no web browsers) as ONE seamless demo
    a. **NOTE:** Essentially, after the slides, I want you to click through and demo of all your upgrades in a mobile emulator/phone BEFORE showing any code
    b. By seamless demo, it means that your features should be **integrated into a single code-base and working app**; points will be deducted for switching emulators/computers/IDEs/branches/etc.
3. **(~5 mins) Code Walkthrough:** Show code and explain key features, code and concepts of how the upgrades work and were implemented
    a. **NOTE:** It is okay to re-open and re-show parts of the app in this section, but your "primary flutter upgrade demo" should already be completed by this point

Other Requirements & Details

GenAI Policy

GenAI chatbots (e.g., ChatGPT, Gemini, etc.) may be used to aid in general strategy (e.g., "My app can currently collect a date from the user, what are a few strategies for displaying dates on a calendar?", "What is a good calendar package?").

GenAI code completion tools like GitHub CoPilot MAY be used **only while in their in-line mode** that turns "comments" into small-to-medium-sized blocks of code, or that automatically comments code when you type "// ", for instance. However, tools like Cline or GitHub Co-Pilot in Agent Mode may NOT be used; to be clear, the distinction that makes these tools (and any others like them) unacceptable is in their ability to create and modify entire files and implement features across multiple files within a project with a single prompt or side-chat built into your IDE. Toward this end, IDEs like Windsurf or Cursor may NOT be used as they tend to have these Agentic features built in too deeply to avoid their usage.

Whenever GenAI code completion tools are used (according to the policy), you should be sure to read the code to ensure you understand it AND because these tools are OFTEN wrong.

NOTE: There is a time and place for these more advanced agentic tools, but it is imperative you first learn the foundations of mobile app development before trying to steer the more advanced tools.

Collaboration

Students should work in their existing group (which may be as an individual, for some).

For both students to get credit, **each student MUST be heard and participate in the submitted demo and must clearly demonstrate that a significant amount of work was put into the upgrades.**

Regardless of what is claimed on the breakdown of labor slide, **a student who does not contribute to the presentation being submitted will receive a 0 on the assignment**.

Attendance

Significant time is given in class to work with your partner and/or ask questions from your peers, professor and tutor. Thus, **attendance is required and will contribute to your grade during designated "lab days"**. **Attendance will be taken at the end of class** and anything short of a documented medical issue will not be accepted.

Grading

See EXCEL rubric alongside this document for grading criteria.